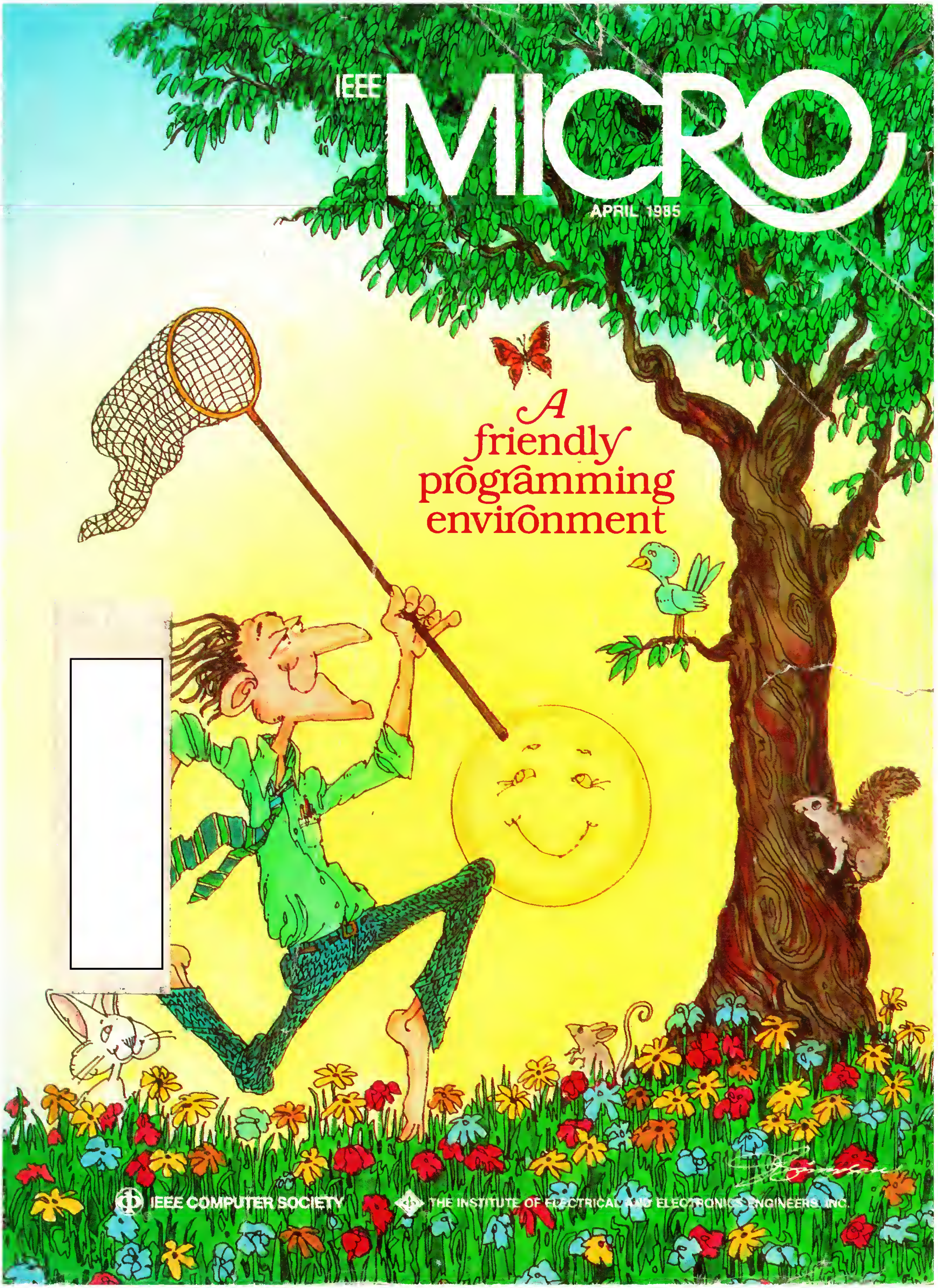

IEEE MICRO
APRIL 1985
A friendly programming environment
IEEE COMPUTER SOCIETY
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE MICRO

April 1985

Volume 5 Number 2(ISSN 0272-1732)

ARTICLES

Cover: A Friendly Programming Environment—see article by Neng F. Yao, page 9.

Cover art: Jay Simpson.

DEPARTMENTS

Reader service cards, p. 113.
Change of address, p. 112.

IEEE Micro

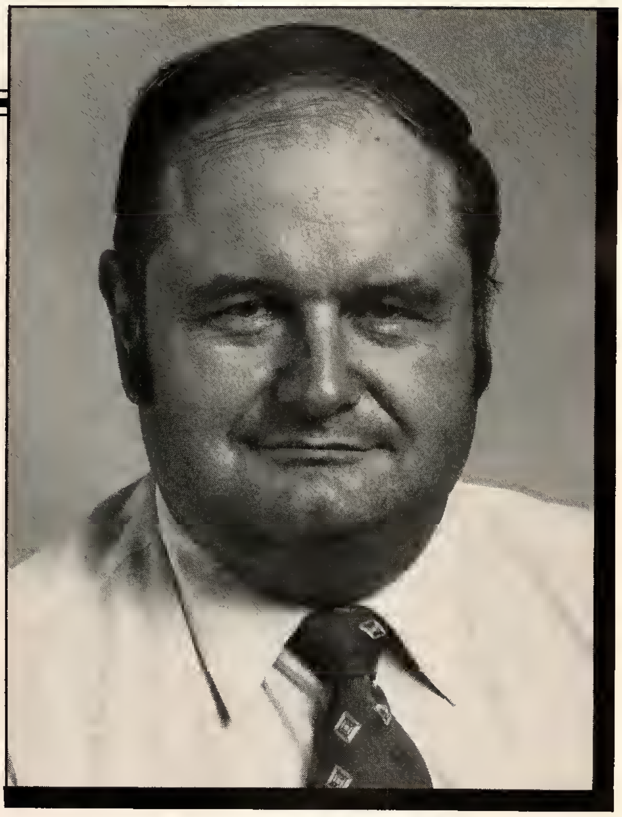
*Submit six copies of all articles and special issue proposals to James J. Farrell III, 1501 Woodhill Drive, Round Rock, TX 78681; (512) 928-6572.

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society: IEEE Headquarters, 345 East 47th St., New York, NY 10017; IEEE Computer Society West Coast Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720. Annual subscription: $12.00 in addition to IEEE Computer Society or any other IEEE society member dues. Nonmember prices: available on request. Single-copy prices: members $7.50; nonmembers $15.00. This journal is also available in microfiche form.

Undelivered copies: Send to 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Postmaster: Send address changes to *IEEE Micro*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854. Second class postage is paid at New York, NY, and at additional mailing offices.

FROM THE EDITOR-IN-CHIEF



Note: Jim Farrell, Associate Editor for the past year on IEEE Micro, *becomes Editor-in-Chief effective April 1, 1985 for a two-year term. Peter Rony, who has completed his two-year term, will continue with* IEEE Micro *as Editorial Consultant.*

First, I would like to thank the Computer Society's Governing Board, Publications Board and Magazine Advisory Committee for my nomination and confirmation as Editor-in-Chief of *IEEE Micro.* I will do my best to continue and enhance the fine work done by Peter Rony, Dick Jaeger, True Seaborn, Joe Schallan, and the rest of the full-time and volunteer professional staff listed on the current and previous mastheads.

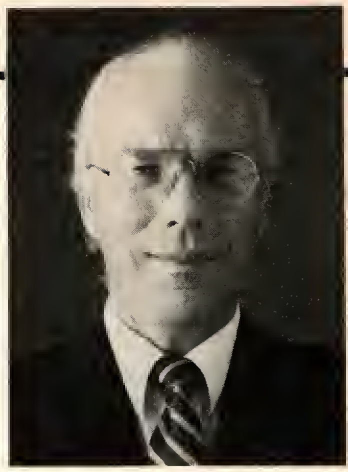
There are many fine electronics magazines in our industry, both from the IEEE and from commercial publishers. Like me, I am sure that most of you subscribe to and read some of these. *IEEE Micro* is different. We are a niche magazine. We are an educational and tutorial magazine that takes both an academic and industrial viewpoint of microprocessors, microcomputers, associated peripheral hardware, software and other related issues. Whether you are still in college or your degree is thirty years old, learning about micros must be a continuing process in this fast moving technology. While the volunteer staff of *IEEE Micro,* including me, is unpaid, we work for you, to get you the finest and most educational information that we can. This is a team effort, and we need you on the team. If you want to point out an error, send a compliment, send a complaint, or have a good idea for *IEEE Micro,* please write to me. This is *your* magazine.

I am aggressively enthusiastic about *IEEE Micro.* I firmly believe that we are involved in one of the most exciting and interesting aspects of electronics. Starting with this issue, we are beginning a new feature, which will list seminars and tutorials where you can learn more about our vocation. If you like it, spend 22 cents and let me know. Would you like to grade the contributed technical articles and respond on the "bingo" (circle number) cards? Write me. Do you have a really great idea for *IEEE Micro?* Send me an outline. Our referees tend to be thorough but fair. Industrial contributed articles will not be refereed by editors from the author's company or a competing company. Conversely, articles submitted by academic authors will be reviewed by industrial editors. My assistant editor-in-chief will be from the academic community and will have the full powers of editor-in-chief when handling industrial contributed articles.

Since we presently have only six issues a year, most of our 1985 calendar queue is pretty full. However, we are considering a worldwide 32-bit micro issue in late 1985 or early 1986. Of course a good article on any micro-related subject will always be well received. In the long term, over the next two years, I would like to see *IEEE Micro* at least double in paid circulation, and become a monthly. If you have an article idea, or an idea of another kind to enhance the magazine, please write to me.

I look forward to hearing from you.

Jim Farrell
1505 Woodhill Drive
Round Rock, TX 78681



microVIEW

Electronic data communications privacy

by Senator Patrick Leahy

Communications technology has left communications law where Einstein's insights left Newtonian physics—tattered and incomplete.

Not long ago, a message was transmitted by first class mail, by wire, or by some form of wireless communications link. Each had its advantages and vulnerabilities. Each was regulated by separate legislation that gave appropriate privacy protection to the user. It was a neat and tidy world, in which private users, common carriers, and government each knew their rights and limits.

The changes in the last 20 years have questioned virtually every assumption of the old universe:

Paper in, paper out. The pre-electronic age was a paper age. Records and communications, however rationally and efficiently kept or transmitted, usually began and ended with a piece of paper. Today, information maintained by business or government is likely to be found on an electronic storage medium, accessible only by a computer.

Where's the data? In the past, information was almost always located in homes and offices. Today, it is equally likely to be found in the hands of a third party custodian, i.e., banks, credit card companies, or electronic mail services.

Where's the alphabet? Another major change is the form in which information is transmitted. Telephones were strictly for the transmission of the voice, and wiretap laws were based on that fact. Today, a large volume of information is converted into digital form—the language of computers—for storage and transmission.

The converging network. Not long ago, most electronic communications relied on common carriers—telephone and telegraph companies. The systems for communicating that information were separate and regulated individually.

They are now converging into an interlocking network of broadcast, telephone, and cable communications. One message can begin by wire, pass through a microwave link, beam up to a satellite, continue on a private fiber-optic connection, and reach its destination via the local phone company.

Enhancing the message. From the date the message was carried at the Battle of Marathon until a scant decade ago, senders and carriers had distinct roles. Carriers did not get involved in the message carried. Today, a service vendor can perform a wide range of services for its communications customer. It can store, analyze, and transform incoming data with the aid of specialized computer programs and then hold the information, return it, transmit it to another point within the user organization, or dispatch parts of it to customers of the user or to others. There are—literally—no limits to the services that can be offered or the nature of the particular relationships that vendors and users will establish.

The laws protecting the privacy of electronic communications, though sensible when enacted, no longer describe a coherent policy that responds to the needs and demands of modern technology.

For example, the basic wiretap law, Title III of the Omnibus Crime Control and Safe Streets Act of 1968, provides criminal sanctions against the interception of wire communications and regulates government wiretaps.

The problem is that when the wiretap law was adopted in 1968, there was little if any focus on the problem of data transmission, and the statute simply fails to cover the interception of data. Other laws, such as the Foreign Intelligence Surveillance Act and the Federal Communications Act, similarly do not provide adequate legal protection against the unauthorized interception of data by government officials or private parties.

New statutes for new problems. No one believes the existing federal laws address the reality of communications technology that has completely outrun old concepts.

We can try to patch up the old laws where the need is pressing. I have proposed amending Title III to cover data and video as well as voice transmissions. But that solution would only deal with common carrier transmissions and not with information handled or maintained by private networks.

Beyond patching up laws urgently needing repair, I want to deal comprehensively with the problems of data communications and privacy in the new information age.

I have been poring over new legislation, and while the form of such a bill is a long way from final, I want to share some of my initial ideas.

First, let me mention a number of distinctions in present laws that no longer serve a useful purpose:

- The law ought to address one central, connecting concept—which I will for the moment call "electronic data communication." It would encompass the sending or receiving of any kind of information in any form, and by any means. EDC would cover the transmission of digitized data by telephone or the transmission of videotex by microwave or any other conceivable mix of medium and message.
- The new law ought to recognize that private carriers and common carriers perform so many of the same functions today that the distinction no longer serves to justify different privacy standards.
- Instead of different laws to cover different modes of electronic communication, there ought to be one principle covering wire communication, radio, microwave, satellite relay, and any other variant that technology might bring us tomorrow.
- Instead of laws offering different levels of (or conditions to) privacy protection based on the mode of transmission or the form of the data, there ought to be a single set of rules that: (1) respect, and by analogy, apply the existing principles of privacy law; (2) consider the reasonable privacy expectations of users of functionally distinguisable services; and (3) balance the competing needs and interests of users and providers.

But while we sweep away old distinctions that no longer make sense, we need to be sensitive to some important new distinctions.

- I want a bill to distinguish between the information contained within a user's (i.e., a sender's) data communication, and files created and maintained by a provider that contain information about a user. Mishandling of both kinds of information threatens the privacy rights of users, but the duties of the providers and the rights of users as to each will be necessarily different.
- The bill should distinguish between transmissions that are expected to be accessed by the public, like electronic bulletin boards, and other data communications.
- Legal remedies should distinguish between intercepts that are done for commercial gain and those that are not, with stronger penalties in the case of the former.

I have not attempted to exhaust the list of topics that will come up when this bill is scrutinized by the Senate.

In fact I am still working on that list, and I hope that by the time I am ready to introduce my bill, I will have the comments and suggestions of many of the IEEE members. You were indispensable to our deliberations last year on the computer chip protection and joint research and development legislation.

I see our job as more than the fine-tuning of existing laws. Electronic mail and other modes of data communication have altered the way this country does business. The changes are fundamental.

At the beginning of our history, first class mail was the chief mode of communication, and throughout the last two centuries it has enjoyed the confidence of the American people and a reputation for preserving privacy, as well as promoting commerce.

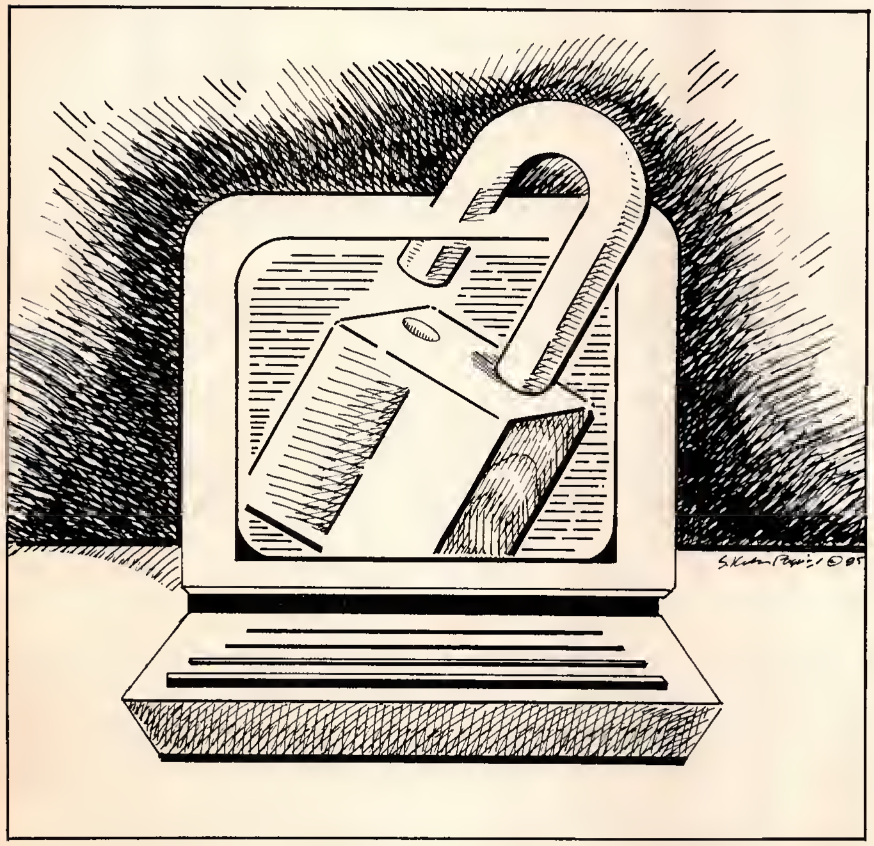
Both of these important goals must continue into our new information age, and we cannot let any American feel less confident in putting information into an electronic mail network than he would putting it into an envelope and dropping it off at the post office.

Your industries have produced a marvelous array of possibilities for better and faster communication. And I know there's more to come.

Now, you and the nation's lawmakers must provide the legal structure and support to allow these possibilities to flourish.

I have no doubt that privacy and progress can be made compatible.

Our job now is to make that promise a reality, and it is a job that you in industry and all of us in Congress must pursue together.



Patrick Leahy of Burlington, Vermont, was elected to the United States Senate in 1974, the first Democrat in Vermont to attain that office. He was 34 and became the youngest Vermonter ever elected to the Senate. He was re-elected to a second term on November 5, 1980.

Senator Leahy is a native of Montpelier, the state capital. He graduated from Saint Michael's College in Winooski, Vermont in 1961, and earned his law degree at Georgetown University Law Center in 1964.

Senator Leahy came to the Senate after serving as State's Attorney for eight years in Vermont's most populous region, Chittenden County. In 1974 he was selected as one of the three outstanding prosecutors in the United States.

He serves on four major Senate Committees: Appropriations, Agriculture, Judiciary and Intelligence. On the Judiciary Committee he serves on the Subcommittee on Patents, Copyrights and Trademarks. In the 98th Congress he played a major role in the successful passage of legislation allowing joint research and development ventures and providing protection for the creators of new semiconductor chips.

LETTERS TO THE EDITOR

To the Editor:

Someone has done the IEEE, its members and its leadership a real disservice. In the August 1984 issue of *IEEE Micro* (p. 4) is an editorial box containing the following phrase: "...Will RAB and TAB still be at each other's throats?..." This unsigned statement is a most misleading observation about the relationship between the entities in the IEEE which represent the geographical and technical interests of the members. We categorically deny any major disputes or animosities between these two major Boards of the Institute. In fact, in several actions of note a warm and cooperative spirit among many of the regional and technical entities exists. Several recent actions are typical:

—TAB's initiative to enlarge its representation on the Board of Directors to enable its Societies to group themselves into technically coherent Divisions would not have succeeded without the support of RAB Directors which was required to achieve the two-thirds majority needed for Board approval.

—RAB's request for increased funding for Sections was fully supported by TAB's Division Directors on the Board of Directors during the most recent budgeting process of Institute funds.

—RAB's wish to have more autonomy in distributing these funds was unanimously supported by TAB whose stated position was that such distribution should be strictly a matter for the RAB Vice President and his Board.

—Current attempts to provide more support for Society Chapters within the Section structure in Regions is being fully supported by both RAB and TAB Directors.

—Extensive and successful efforts by the Computer Society's Division Director Merlin Smith, newly elected Vice President of the IEEE, to work with both Boards in the area of conferences.

—Most cordial relationships between RAB and TAB Directors in numerous minor matters which come before the Board.

—Initiatives by both the undersigned to examine any points of potential dispute through a RAB/TAB Liaison Committee under the joint leadership of TAB Director Harold Sorenson and former RAB Membership Development Committee Chairman Roy Yee.

There are always issues to be discussed and compromise positions to be taken between interested parties working in good faith for a common purpose. To characterize them as "at each others throats" is misleading and counterproductive. It is certainly not in the best interests of the world's largest professional technical organization nor its largest and very successful Computer Society. It is most disappointing that the anonymous writer of the editorial did not find it necessary to check with either of us, the chairmen of the affected RAB and TAB Boards, as to our perceptions of the RAB/TAB relationships. We deeply regret the insensitivity and ignorance of the facts which is implied by the comment and assure the readers of this magazine that there is no truth to the assertion.

Merrill Buckley
1984 IEEE Vice President
Regional Activities

Stephen Kahne
1984 IEEE Vice President
Technical Activities

The author responds:

I am indeed miffed that the act of placing printed material in a box was interpreted to mean that the author is anonymous. As the author of the boxed section I accept full credit, and blame, for it. I am indeed happy to hear that the RAB (Regional Activities Board) and TAB (Technical Activities Board) are now working harmoniously together. It would be wrong to say that situation has always prevailed in the past. In fact, Charles (Bud) Eldon, now President of the IEEE, in his address to the TAB after lunch at the meeting recently in Houston on February 22, explicitly referred to the acrimonious relationships that had existed in the past between the RAB and TAB directors. He too was pleased that situation seems now to have changed.

It is clear that an organization like the IEEE is not always perfect. It must make its way amongst the rock piles of life as do we all. The inspiring aspect I've observed is the tremendous labors of love its dedicated volunteers contribute to the organization. That's why I finished the boxed segment with:

"May the IEEE adapt and grow and continue to be a great organization undertaking great tasks for a great profession."

Robert G. Stewart
First Vice President
IEEE Computer Society

To the Editor:

The June 1983 issue of *IEEE Micro* (pp. 40-50) carried the excellent article by A. Perez entitled, "Byte-wise CRC Calculations." The super-fast approach presented used the Perez CRC Table lookup algorithm for the Bisync protocol that uses the generating polynomial of X +X X +1, also called CRC-16.

Using Mr. Perez' brilliant approach we derived a table for the IBM SDLC, CCITT HDLC, and X.25 protocols, which utilize a generating polynomial of X +X +X +1. The CRC dividend is handled somewhat differently by these protocols. The subroutine in Figure 1 illustrates these differences. In addition to the newly generated lookup table values, the dividend/remainder is initialized to all 1's, the two CRC bytes transmitted are an inversion or ones-complement of the generated CRC value, the high bit of the CRC 2 byte field is transmitted first, and the dividend/remainder of a correctly received frame yields the constant 61624 decimal.

We have used Mr. Perez' super-fast CRC algorithm with the Richcraft SDLC, HDLC, and X.25 software approach to packet communications with great success the past year. Our software approach completely eliminates the expensive terminal node controller on all nodes running the Bell 202 type standard, synchronously at 1200 baud and up.

Robert M. Richardson
Richcraft Engineering Ltd.
Chautauqua, NY 14722

Figure 1a. Modified lookup table for byte-wise CRC calculations.

```
00100 ;
00110
00120 ; PEREZ CRC GENERATION/CHECKING LDDK-UP TABLE APPROACH
0D130 ; MODIFIED FOR IBM SOLC, CCITT HOLC & X.25 PROTOCOLS.
00140
00150 ; TCRC IN LINE 150 GENERATES THE CRC 2 BYTE VALUE FOR A
00160 ; FRAME BEGINNING AT (AOOREZ) IN MEMORY AND OF 'BC' BYTES
0017D ; LENGTH.
0D180
0019D ; FINCRC IN LINE 580 INVERTS THE TWO CRC BYTES AND LOAOS
0D2D0 ; THEM INTO MEMORY IN THE CORRECT ORDER.
00210
DD220 ; RCRC BEGINNING IN LINE 70D TESTS THE RECEIVED CRC VALUE
00230 ; OF A FRAME STARTING AT (BGINIT) IN MEMDRY WITH A TOTAL
00240 ; LENGTH OF (LENG1) BYTES.
0D25D
0026D ; TABLE BEGINNING AT LINE 85D IS THE MDDIFIED VERSION.
DD27D
00280 CRCVAL  OEFW  0              ;RECEIVE CRC VALUE STASH
DD29D ENDCRC  DEFW  0              ;XMIT CRC VALUE STASH
DD300 TCRC    LD    HL,(AOOREZ)    ;BEGIN FRAME MEM LOCATION
0031D         LD    BC,(LENG1)     ;LENGTH DF FRAME IN BYTES
DD320         LO    DE,65535       ;INITIALIZE DIVIDENO 1'S
0D33D         CALL  DOCRC          ;GENERATE FRAME CRC
0D34D         CALL  FINCRC         ;SORT/STUFF RIGHT ORDER
00350         RET                  ;RETURN WHENCE U CAME +1
DD36D DOCRC   LD    A,(HL)         ;FIRST BYTE TO CRC
00370         INC   HL             ;INCREMENT FOR NEXT DNE
DD380         PUSH  BC             ;SAVE BYTES TO CRC
DD39D         PUSH  HL             ;SAVE NEXT BYTE LOCATIDN
00400         XOR   E              ;XDR REMAINDER LSB W/'A'
DD410         LD    C,A            ;SAVE RESULT IN 'C'
00420         LD    B,0            ;ZERO OUT 'B'
DD43D         LO    HL,TABLE       ;LDDKUP TABLE LDCATIDN
DD44D         ADD   HL,BC          ;ADD BC TD LDCATIDN
DD45D         ADD   HL,BC          ;ADD BC TD LDCATIDN
0046D         LD    A,D            ;REMAINDER MSB TD 'A'
D047D         XOR   (HL)           ;XOR WITH TABLE VALUE
00480         LO    E,A            ;LDW BYTE CRC IN 'E'
DD490         INC   HL             ;NEXT TABLE LOCATION
DD5DD         LD    0,(HL)         ;HIGH BYTE CRC IN 'D'
D051D         POP   HL             ;NEXT BYTE TD CRC MEM
DD520         POP   BC             ;NUMBER BYTES TO CRC
DD53D         DEC   BC             ;LESS DNE
0054D         LD    A,B            ;TEST FOR
DD55D         DR    C              ;ZERO
0056D         JP    NZ,DOCRC       ;IF NOT, CRC NEXT ONE
D0570         RET                  ;ELSE ALL DDNE. RETURN
DD58D FINCRC  LD    A,E            ;DE = CRC 2 BYTE VALUE
DD59D         CPL                  ;COMPLEMENT IT
006DD         LO    HL,(WHER4B)    ;ENO DF FRAME IN MEM +1
00610         LD    (HL),A         ;LD 1ST CRC BYTE IN MEM
DD62D         LO    (ENDCRC+1),A   ;AND SAVE IT HERE
00630         INC   HL             ;NEXT MEM LOCATION
DD64D         LO    A,D            ;SECOND CRC BYTE
0065D         CPL                  ;CDMPLEMENT IT
DD660         LO    (HL),A         ;LD 2ND CRC BYTE IN MEM
DD67D         LO    (ENDCRC),A     ;AND SAVE IT HERE
0068D         RET                  ;RETURN WHENCE U CAME +1
00690
D0700 RCRC    LD    DE,65535       ;INITIALIZE DIVIDEND 1'S
DD71D         LD    HL,(BGINIT)    ;BEGIN FRAME LOCATIDN
0072D         CALL  OOCRC          ;CRC ALL INCLUOING CRC
0D73D         LD    (CRCVAL),DE    ;SAVE REMAINOER IN MEM
DD74D         LO    HL,61624       ;COMPARE REMAINDER WITH
DD750         RST   18H            ;61624 OECIMAL
DD76D         JP    NZ,BAOCRC      ;NOT ZERO = BAD ONE
00770         RET                  ;DK, SO RETURN
DD780 BADCRC  POP   AF             ;ADJUST STACK FOR CALL
0079D         LD    IY,37692       ;<BAD CRC> MESSAGE
0080D         CALL  SHOWIT         ;DISPLAY DN VIDED
00810         JP    MDDE1A         ;GD AWAIT NEXT PACKET
DD820
00830 ; MOOIFIED LDOKUP TABLE FDR BYTE-WISE CRC SUBRDUTINE
```

Figure 1b. This is the 512-byte CRC look-up table printed out as 256 two-byte words to save space. The label TABLE is at location 1.

1	DEFW	0	24	DEFW	25662	47	DEFW	51324	70	DEFW	5545	93	DEFW	39145	116	DEFW	16668
2	DEFW	4489	25	DEFW	4D137	48	DEFW	55797	71	DEFW	10034	94	DEFW	35168	117	DEFW	13731
3	DEFW	8978	26	DEFW	36160	49	DEFW	12675	72	DEFW	14011	95	OEFW	48123	118	DEFW	9258
4	DEFW	12955	27	DEFW	49115	50	DEFW	8202	73	DEFW	52812	96	DEFW	43634	119	DEFW	58D9
5	DEFW	17956	28	DEFW	44626	51	DEFW	4753	74	DEFW	57285	97	DEFW	2535D	12D	DEFW	1848
6	DEFW	22445	29	DEFW	56045	52	OEFW	792	75	OEFW	60766	98	DEFW	29327	121	DEFW	65487
7	DEFW	25910	30	DEFW	52068	53	OEFW	30631	76	OEFW	64727	99	DEFW	16404	122	DEFW	60998
8	DEFW	29887	31	DEFW	63999	54	DEFW	26158	77	OEFW	34920	100	OEFW	20893	123	DEFW	56541
9	DEFW	35912	32	DEFW	5951D	55	DEFW	21685	78	DEFW	39393	1D1	DEFW	95D6	124	DEFW	52564
1D	DEFW	4D385	33	DEFW	845D	56	DEFW	17724	79	DEFW	43898	1D2	DEFW	13483	125	DEFW	47595
11	DEFW	44890	34	DEFW	12427	57	DEFW	48587	8D	DEFW	47859	1D3	DEFW	1584	126	DEFW	43106
12	OEFW	48851	35	DEFW	528	58	DEFW	44098	81	DEFW	21125	104	OEFW	6073	127	DEFW	39673
13	DEFW	5182D	36	DEFW	5017	59	DEFW	4D665	82	DEFW	17164	105	OEFW	61262	128	DEFW	35696
14	DEFW	56293	37	DEFW	26406	6D	DEFW	36688	83	DEFW	29079	106	OEFW	65223	129	DEFW	33800
15	OEFW	59774	38	DEFW	30383	61	DEFW	264495	84	DEFW	24606	1D7	OEFW	52316	130	DEFW	38273
16	OEFW	63735	39	DEFW	17460	62	DEFW	60006	85	DEFW	5281	108	OEFW	56789	131	DEFW	42778
17	DEFW	4225	4D	DEFW	21949	63	DEFW	55549	86	DEFW	1320	1D9	DEFW	43370	132	DEFW	46739
18	DEFW	264	41	DEFW	44362	64	DEFW	51572	87	DEFW	14259	11D	DEFW	47331	133	DEFW	49708
19	DEFW	13203	42	DEFW	48323	65	DEFW	16900	88	DEFW	9786	111	OEFW	35448	134	DEFW	54181
2D	OEFW	8730	43	DEFW	36440	66	DEFW	21389	89	DEFW	57037	112	DEFW	39921	135	DEFW	57662
21	DEFW	22181	44	DEFW	40913	67	DEFW	24854	9D	DEFW	53D6D	113	DEFW	29575	136	DEFW	61623
22	DEFW	18220	45	DEFW	6D270	68	DEFW	28831	91	DEFW	64991	114	DEFW	251D2	137	DEFW	2112
23	DEFW	30135	46	DEFW	64231	69	DEFW	1056	92	DEFW	60502	115	DEFW	20629	138	DEFW	6601

139	DEFW	11090	159	DEFW	32247	179	DEFW	38553	199	DEFW	41786	219	DEFW	31191	238	DEFW	15595
140	DEFW	15067	160	DEFW	27774	180	DEFW	34576	200	DEFW	45747	220	DEFW	26718	239	DEFW	3696
141	DEFW	20068	161	DEFW	42250	181	DEFW	62383	201	DEFW	19012	221	DEFW	7393	240	DEFW	8185
142	DEFW	24557	162	DEFW	46211	182	DEFW	57894	202	DEFW	23501	222	DEFW	3432	241	DEFW	63375
143	DEFW	28022	163	DEFW	34328	183	DEFW	53437	203	DEFW	26966	223	DEFW	16371	242	DEFW	58886
144	DEFW	31999	164	DEFW	38801	184	DEFW	49460	204	DEFW	30943	224	DEFW	11898	243	DEFW	54429
145	DEFW	38025	165	DEFW	58158	185	DEFW	14787	205	DEFW	3168	225	DEFW	59150	244	DEFW	50452
146	DEFW	34048	166	DEFW	62119	186	DEFW	10314	206	DEFW	7657	226	DEFW	63111	245	DEFW	45483
147	DEFW	47003	167	DEFW	49212	187	DEFW	6865	207	DEFW	12146	227	DEFW	50204	246	DEFW	40994
148	DEFW	42514	168	DEFW	53685	188	DEFW	2904	208	DEFW	16123	228	DEFW	54677	247	DEFW	37561
149	DEFW	53933	169	DEFW	10562	189	DEFW	32743	209	DEFW	54925	229	DEFW	41258	248	DEFW	33584
150	DEFW	49956	170	DEFW	14539	190	DEFW	28270	210	DEFW	50948	230	DEFW	45219	249	DEFW	31687
151	DEFW	61887	171	DEFW	2640	191	DEFW	23797	211	DEFW	62879	231	DEFW	33336	250	DEFW	27214
152	DEFW	57398	172	DEFW	7129	192	DEFW	19836	212	DEFW	58390	232	DEFW	37809	251	DEFW	22741
153	DEFW	6337	173	DEFW	28518	193	DEFW	50700	213	DEFW	37033	233	DEFW	27462	252	DEFW	18780
154	DEFW	2376	174	DEFW	32495	194	DEFW	55173	214	DEFW	33056	234	DEFW	31439	253	DEFW	15843
155	DEFW	15315	175	DEFW	19572	195	DEFW	58654	215	DEFW	46011	235	DEFW	18516	254	DEFW	11370
156	DEFW	10842	176	DEFW	24061	196	DEFW	62615	216	DEFW	41522	236	DEFW	23005	255	DEFW	7921
157	DEFW	24293	177	DEFW	46475	197	DEFW	32808	217	DEFW	23237	237	DEFW	11618	256	DEFW	3960
158	DEFW	20332	178	DEFW	41986	198	DEFW	37281	218	DEFW	19276						

To the Editor:

I thought your article, "Copy-protection-defeating programs: Should Congress Act?" in the December 1984 *IEEE Micro* was excellent. I wish Congress great wisdom and godspeed in solving the nasty problems of copyright protection in the context of cheap, anonymous, electronic duplication of machine-readable files, including computer programs.

There appear to me to be two areas where the problems will be even nastier than your article indicated.

The first is really outside the scope of your article: with cheap print-reading devices coming on the market, it will cost very little to put almost any text into "machine-readable" form (see the Omni-Reader ad on page 92 of the same issue of *IEEE Micro*, for example). Once there, of course, it is subject to the same cheap, anonymous duplication (and distribution over the telephone network) as are computer programs. The duplication of a book by office copier may well be more expensive than purchase of the book, but the duplication of a machine-readable version of the text is not likely to be so. The technology jeopardizes copyrights on more than just computer programs.

Not everyone will immediately be eager to read a best-seller from a computer console, but the possibility exists. And now there are bona fide text-to-speech systems on the market. (Have you listened to DecTalk lately?)

The second problem area is very much within the scope of your article and is of major concern to many of us already: floppy disks are a very inconvenient medium for storage of programs that are used frequently. Many of us owners of small computers have bought hard disks and/or extra RAM to avoid the clumsy manual operation and low performance that results from use of floppies. Copying a program from the source floppy disk brings benefits to us in some combination of three main ways.

First, if one wants to use a program that is resident only on a floppy disk, that disk must be mounted in a disk drive (thus excluding other uses during program-load time...and during overlay-thrash time, as noted below). Also, if that drive is needed for a data input or output disk associated with the task in hand, someone must be present and alert to swap disks when needed. Further, simple safety procedures require that a floppy (especially one whose contents are not backed-up somewhere) must be removed before the system is powered down. A program that cannot be copied from one floppy to another presumably cannot be copied from a floppy to hard disk, so a copy-protected program requires the dedication of a human being to mind-numbing, error-prone tasks. The cost in dollars and misery is high. This cost adds to the incentives to copy disks, even if protected.

Second, disk transfer speeds from hard disks are higher than from floppies and are likely to remain so. Even initial program load time is noticeably faster for programs on hard disk, and for compilers and other large programs with overlays that act on large files a piece at a time, the improvement in performance with a hard disk is often quite significant. More incentive.

Third, many operating systems provide for dedication of a portion of RAM to be used temporarily as a pseudo-disk (e.g. MDRIVE in MS-DOS v.2.05). Its performance is even higher than that of a hard disk. An application that uses MDRIVE fairly hums, and the user gets strikingly improved response from the computer system. Conversely, after one uses MDRIVE and then is forced to revert to operation from a floppy drive, the pauses often seem interminable. The pseudo-disk disappears when the system is powered down, and any information on it is lost. A copy-protected program can't be transferred to or run on MDRIVE, of course, so here is an additional incentive to use a copy-protection-defeating program.

The excerpt you gave from Section 117 of the Federal Copyright Act requires that, to avoid infringement, a copy must be "...created as an *essential* step in the utilization of the computer program..." (emphasis mine). Use of the hard disk and RAM-resident pseudo-disk are essential only to efficiency (and perhaps survival) of the business, not to utilization of the computer program, so it appears to me that Section 117 doesn't meet some important current needs of a software user. The proposed corrective measures documented in your article seem incomplete to me in the light of the problems I have raised, and I would like to have the measures re-cast before I choose among them.

I believe that any measure that discusses *backup* and *archive* copies only and doesn't address the change-of-medium problems I have presented here will almost surely fail to relieve many law-abiding computer owners of incentives to bypass the copy-protection mechanisms.

So far, I have stayed squeaky-clean with respect to making copies of copy-protected disks. I have done so by refusing to buy software that only comes in that form. I have had to pass up some excellent software that way, of course. Both the software vendors and I are suffering that penalty. As a possible future vendor of software, I have even more reason to hope for a quick, workable, equitable solution to an extremely sticky set of problems.

Let the debate continue, but not forever. Thank you for your contributions.

Roy Keir
530 Northmont Way
Salt Lake City, UT 84103

A conventional system puts the programmer through a time-consuming cycle of manual referencing, code correction, and program re-execution. This system provides a window-based on-line help facility to ease and speed the programming process.

The Computer-aided Programming System—A Friendly Programming Environment

Neng F. Yao

Computers are machines that play an important and unique role in modern technology in that they amplify man's mental rather than his physical power. More and more people are using and relying on computers to help solve problems. However, during the past decade there has been growing awareness of a communication gap between man and computer.[1-4] A nonprofessional computer programmer, or even a professional programmer, usually has to rerun his program many times and spends a great amount of time debugging in order to obtain the desired output. The question of the friendliness of the computer becomes a significant concern in programming environments.[5,6]

There are three alternative solutions to the problem of surmounting the barrier between man and computer[7]:

- teach or train all users to become computer programmers,
- provide all users with their own interpreters or programmers, or
- design the computer to be more friendly.

The first approach is not feasible because of each individual's personal characteristics and educational background, and because of the time it would consume. The second approach is not only too costly but also removes the user from direct control over the computer. An additional communication barrier—between the user and his programmer—is thereby generated.

The Computer-aided Programming System—CAPS—utilizes the third approach; it is an interactive programming environment with integrated facilities to create, edit, execute, and debug programs. In the CAPS, a high-level-language, direct-execution concept is combined with a computer-based guidance feature.[6] With computer-based guidance, not only error messages but also suggestions

0272-1732/85/0400-0009$01.00 © 1985 IEEE

for the next statement are given. A programming environment without these features is defined as a conventional programming system, or CPS.

The section below discusses the hardware and software architecture of the system. The third section of this article compares sample CAPS and CPS programs. The concept of analyzing programs by state diagrams is introduced in the fourth section. The fifth and final section discusses possible applications of, and further research on, the CAPS.

Architecture of the CAPS

Hardware system configuration. The hardware configuration of the CAPS with a computer numerically controlled (CNC) machine tool application is shown in Figure 1. There are seven system units: the SN-1 microcomputer, an input keyboard, an editing CRT terminal, a graphics CRT terminal, an output printer, a numerically controlled machine tool, and a system panel.

The SN-1 microcomputer is the core of the system. It is a microprogrammable microcomputer based on the Am2903 bit-slice microprocessor.[8,9] The address bus and the data bus of the system are both 32 bits wide. The microinstruction word is 96 bits wide.

The user enters his program through the keyboard. The program text, error messages, and guidance messages are displayed on the editing CRT terminal. While the program is being created, edited, and executed, the physical movements of the numerically controlled machine tool can be simulated on the graphics CRT terminal. After

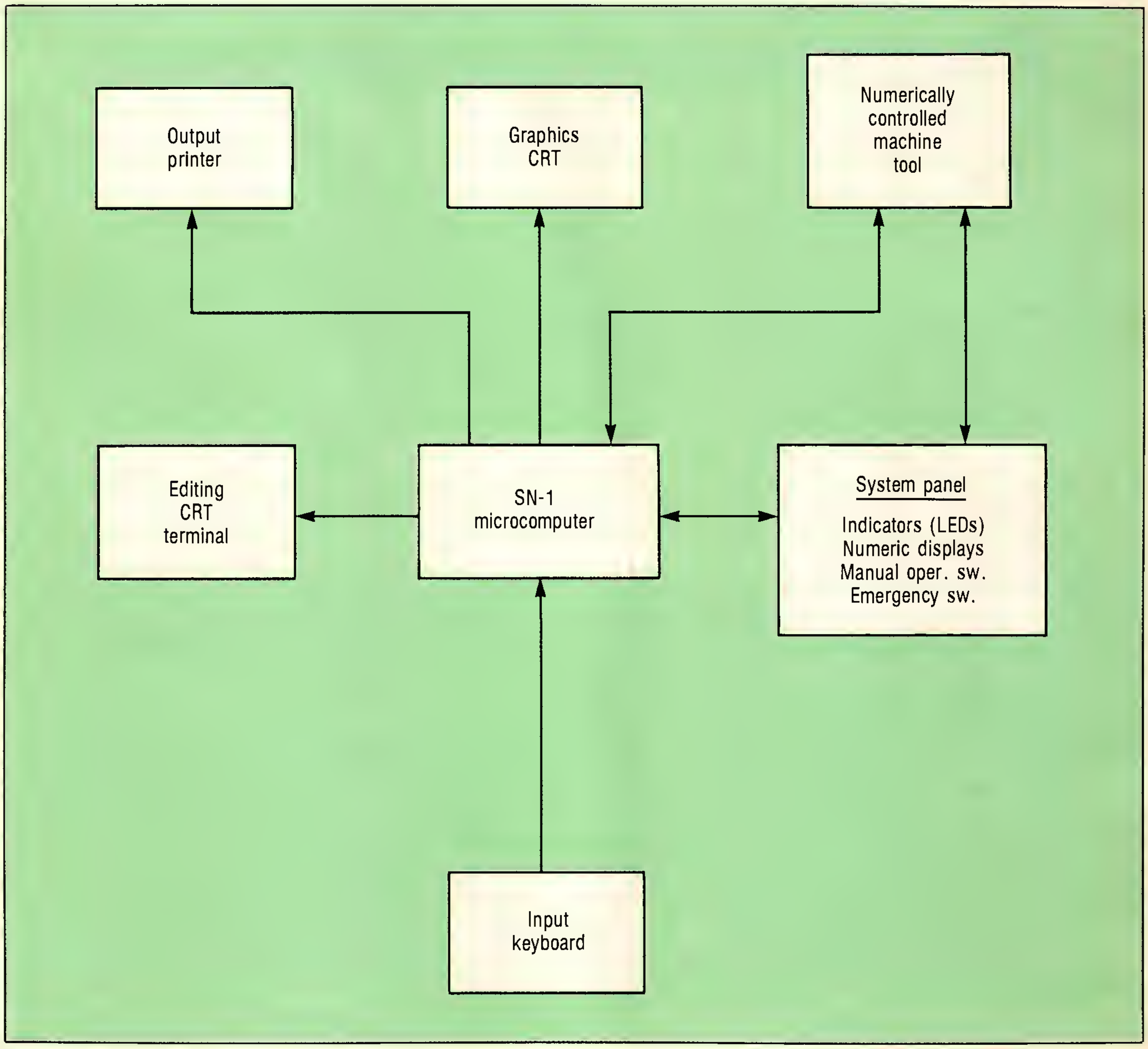


Figure 1. Hardware configuration—the CAPS with a CNC application.

the desired results have been obtained on the graphics CRT, the output of the program can be shifted to the machine tool to produce parts.

The system panel provides LED indicators and numeric displays to show the status of the system. Switches are also provided for manual and emergency operation.

Software system configuration. Computer-assisted programming systems have been restricted to syntax error diagnosis.[10-13] This is a passive approach, since the system gives error messages only after an incorrect statement has been entered. Suppose the user does not know the name of the function or the syntax of the next statement—in a conventional environment, he can use the trial-and-error method, or terminate the whole programming process to seek help (e.g., look at manuals). Both of these strategies are cumbersome and unproductive. The CAPS provides assistance in an active way. It not only diagnoses the error after it has been made, but also provides guidance as soon as the user asks for help or for directions for the next step.

The system's software configuration can be described with a flowchart (Figure 2). First, subroutine INITIALIZATION initializes flags, tables, stacks, registers, and buffers to the proper values for the editor, the translator, and the interpreter. Subroutine EDITOR, which is a full-screen-type editor,[14] accepts text characters and control commands from the keyboard terminal. Users modify their programs by using cursor movement functions (e.g., MOVE, INSERT, and DELETE keys). After the end-of-statement character has been entered, control shifts to subroutine TRANSLATOR. The translator translates the current source statement, which is written in high-level language, into intermediate codes (ICODE). If, in the process of translation, a syntax or logic error is detected, a meaningful error message is produced and control of the system goes back to the editor. If guidance is requested, a guidance message is displayed to help the user with the next statement.

Subroutine INTERPRETER then executes the ICODE and sends any result to the output device. In graphics simulation, the user sees the result on the output terminal after each statement has been entered. Control then shifts to the editing mode, and the system waits for the next source statement or editing command. The user can then examine these results and choose either to continue with the next statement or to change one or more of the statements previously entered by employing the full-screen editor. After the statement has been altered, it will immediately be retranslated and reexecuted by the system automatically. The time and effort required by the conventional system to shift from editor to translator, translator to interpreter, and back to editor again are eliminated. Therefore, syntax and logic errors are easier to locate and correct. Once a program has been entered and executed, its intermediate codes are generated and stored for future executions. The editing and translating can be skipped, as shown in the flowchart, provided that the ICODE file has been generated previously. There are flags which can be set or reset to vary the flow of the system. For instance, if the program execution flag is reset, then the program will be edited and translated only. Execution will be skipped and no result will be generated. The trap flag is provided so the user can set a breakpoint in a program. If the single-step option is selected, the execution of each statement will be halted until a GO command is released.

Sample programs and comparisons

The objective of our sample program is to draw the figure shown in Figure 3 on a graphics terminal. The source language is LNCM, the Language for Numerically Controlled Machines, which is a high-level language that has been standardized by the Electronic Industries Association.[15] A summary of the definitions for LNCM is given in Appendix A. The correct program is as follows:

```
%;     PROGRAM START
N100   G92    X0      Y8;      START POINT A
N110   G01    X0      Y-6;     LINE A TO B
N120   G12    R2      C180;    DEFINE RADIUS & CENTER
N130   G11    C180;            ARC B TO C, CCW
N140   G01    X0      Y6;      LINE C TO D
N150   G01    X-4     Y0;      LINE D TO A
N160   M02;                    FINISH
```

A statement in the program is also defined as a block.

Assume the user makes the following errors in his first attempt at creating and executing the program:

- misses or does not know of the need for the start block (%;) (line 1),
- forgets the function number G01 for drawing line segments (lines 3, 6, and 7),
- misses the definition block (G12 . . .) for the radius and the center of the arc (line 4),
- uses the wrong function (G10) to draw a counterclockwise arc (line 5), and
- erroneously types the digit "0" as the letter "o" for the Y parameter (line 7).

Mistakes such as the above are commonly found in high-level-language programs. Some of them are syntax errors and some are logic errors.

Programming under a CPS. In a conventional programming system, the user first creates the program with an editor; let us assume the program incorporates the errors discussed above:

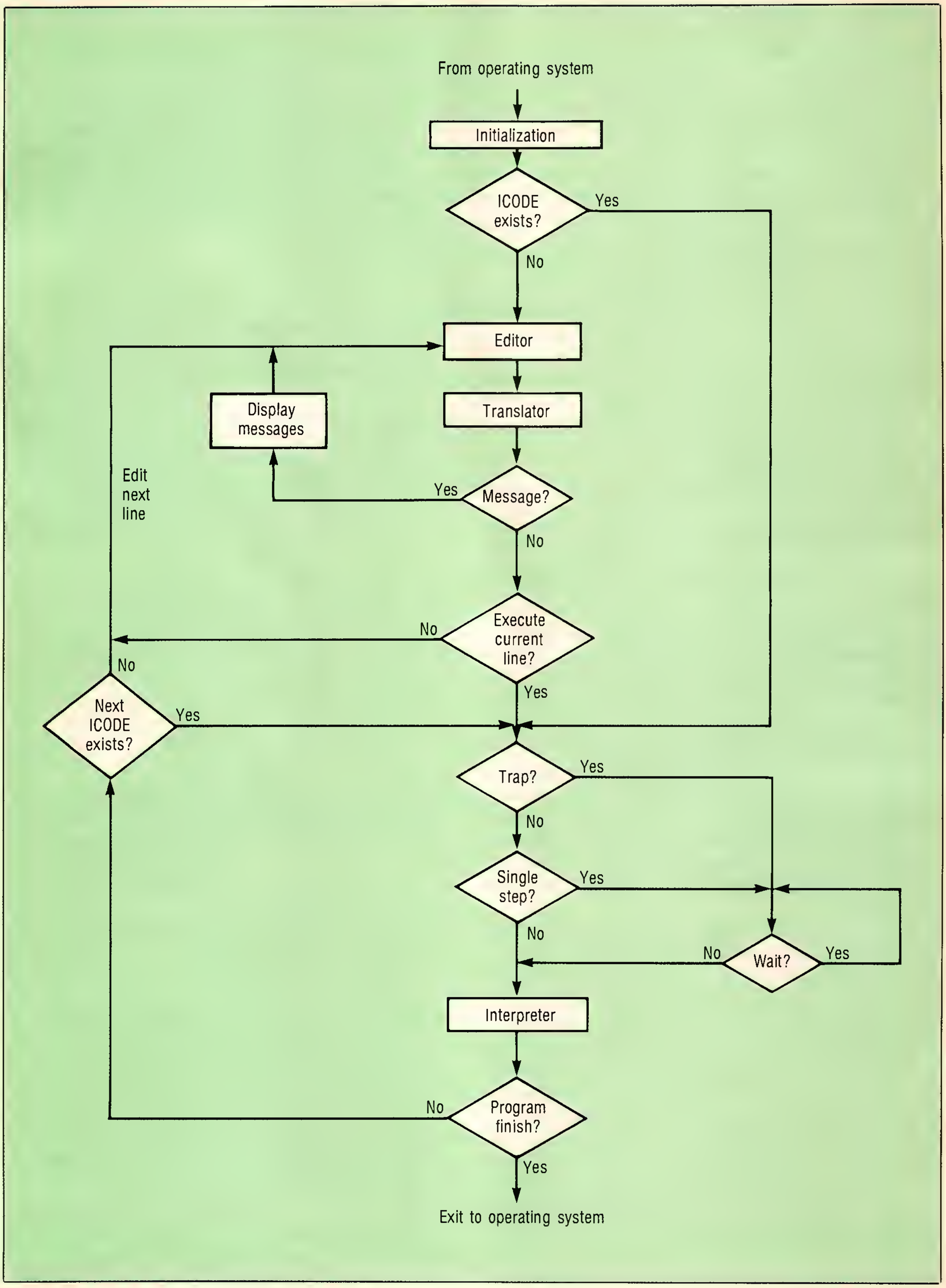


Figure 2. Flowchart for the CAPS.

```
        PROGRAM START
N100    G92    X0     Y8;     START POINT A
N110    G?     X0     Y-6;    LINE A TO B
N130    G10    C180;          ARC B TO C, CCW
N140    G?     X0     Y6;     LINE C TO D
N150    G?     X-4    Yo;     LINE D TO A
N160    M02;                  FINISH
```

After the program has been translated, an error message appears:

```
*** ERROR NO. 1 IN LINE(S) 1
```

The user then has to check with the reference manual to find out what this means. After finding out that the program start block (%;) has been missed, he has to go back to the editor and make the correction. Then the program is ready to be translated again. This time, the error message is

```
*** ERROR NO. 2 IN LINE(S) 3, 5, 6
```

Now the user has to check the reference manual again to find the function number for drawing line segments. After reediting and retranslating the program, he gets another error message:

```
*** ERROR NO. 3 IN LINE(S) 6
```

Checking the manual yet again and carefully examining the characters in line 6, he finds out that the digit "0" following the parameter Y has been erroneously typed as the letter "o". After using the editor four times, the reference manual three times, and the translator four times, the user now has a program free of syntax errors:

```
%;      PROGRAM START
N100    G92    X0     Y8;     START POINT A
N110    G01    X0     Y-6;    LINE A TO B
N130    G10    C180;          ARC B TO C, CCW
N140    G01    X0     Y6;     LINE C TO D
N150    G01    X-4    Y0;     LINE D TO A
N160    M02;                  FINISH
```

An intermediate code file—sometimes called an object code file—can now be generated by the translator and executed.

After the intermediate codes are executed, however, the user gets an erroneous output (Figure 4). The line segment AB is correctly drawn, but the desired semicircle does not appear. This is a typical logic error, and it represents the type of error that conventional programming systems have the hardest time handling. To find an algorithmic or logical error, the user has to spend a great deal of time analyzing the program and checking the reference manual. He gets no assistance from the computer, since a conventional programming system can only provide syntax diagnosis. The error here is that the radius and the center of the arc have been left undefined.

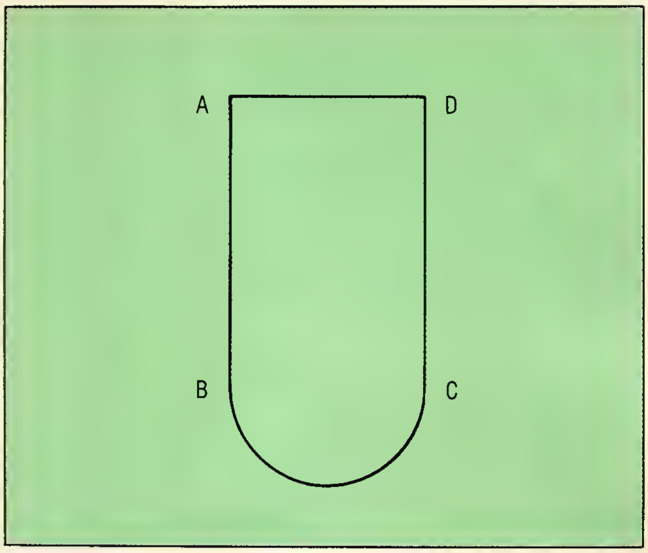


Figure 3. The figure to be drawn by the LNCM program.

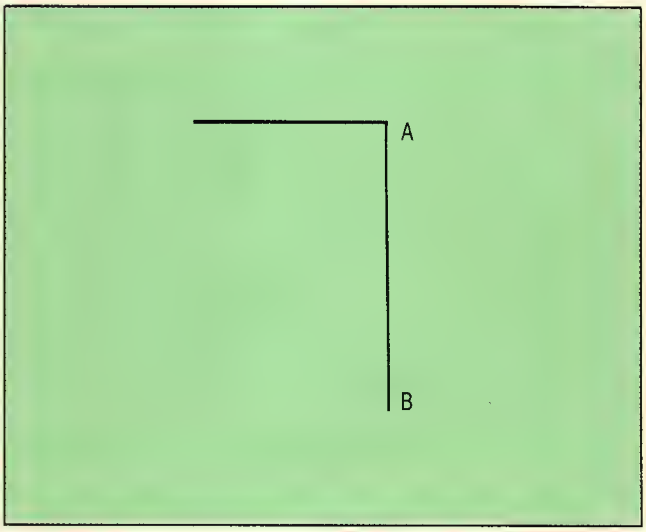


Figure 4. Erroneous output from the executed LNCM program.

Therefore, the user has to return to the editor again to add the statement for the radius and the center:

```
N120    G12    R2     C180;   DEFINE RADIUS & CENTER
```

Another translation and execution of the program produces the result shown in Figure 5. A clockwise semicircle has been drawn from B to C instead of a counterclockwise one. Now the user has to refer to the manual again to find the function that draws arcs in a counterclockwise direction. And then he has to go through editing, translating, and executing the program once more to obtain the correct source program and the desired output.

Programming under the CAPS. In the computer-aided programming system, the CRT screen is divided into two

windows. Editing is performed in the upper window and guidance and help messages (Appendix B) are displayed in the lower one. The number of text lines reserved for each window is user-programmable.

The help messages (Appendix B.2) provide language definition information. While creating or editing a program, the user can request language syntax or definition information using the commands shown in the message window. The first line in the message window continually displays the following:

```
HELP DIR: ENTER CONTROL Z,  EXIT HELP: ENTER CONTROL W
```

The principle of the CAPS is as follows. Initially, the CAPS is in the editing mode, waiting for an input line. When an end-of-line (EOL) is encountered, the system automatically shifts to the translating mode. If no syntax error is detected, the system shifts directly to the executing mode and generates a result, if there is any. After this, the CAPS transfers back to the editing mode and waits for the next input line. If errors are detected in the translation phase, messages are displayed in the message window (lower part of the screen).

If the function field in the input statement is neglected, the CAPS assumes that the user does not know the function number for the desired operation. The system then analyzes the statements inputted previously and suggests functions via guidance messages. The execution phase is skipped. Then the CAPS shifts back to the editing phase and waits for the user to complete the statement.

In order to obtain a comparison, let us examine the process of creating the sample LNCM program under the CAPs. We assume that the user makes the same mistakes as in the previous sample.

The user first enters a source statement in the editing window:

```
PROGRAM START
```

The CAPS then displays the following message in the message window:

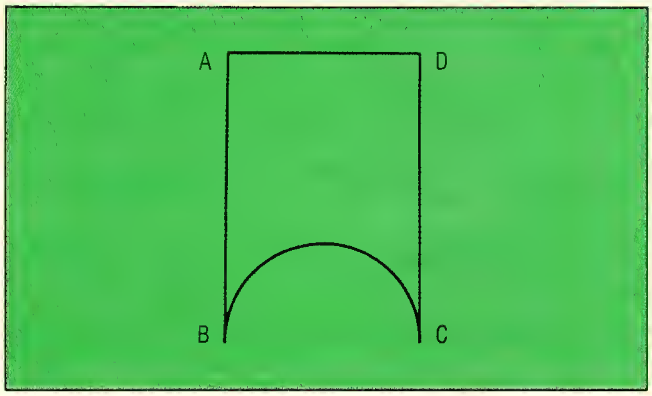


Figure 5. Output figure with semicircle facing in wrong direction.

```
**ENTER PROGRAM START BLOCK "%;"
```

It then automatically shifts back to the editing mode and waits for the user to enter the start block. The user then easily changes the first line to

```
%;     PROGRAM START
```

Since there is no syntax or logic error in this altered statement, the CAPS translates it, executes it, and then shifts back to the editor and waits for the next statement, without displaying messages. The user now enters the next statement:

```
N100   G92   X0     Y8;     START POINT A
```

This block is translated and executed without an error. The starting point of the line segment AB is drawn on the output screen. The CAPS then shifts to the editing mode. The user now enters the sequence number N110, but he does not know the function number for drawing a line segment from point A to point B. Instead of randomly guessing or referencing the manual, however, he simply enters an EOL command to ask for the computer's guidance. In response, the following is displayed:

```
%;     PROGRAM START
N100   G92   X0     Y8;     START POINT A
N110

HELP DIR: ENTER CONTROL Z, EXIT HELP: ENTER CONTROL W
**ENTER G92: DEFINE START POINT OR
        G12: DEFINE CENTER & RADIUS FOR ARC OR
        G01: DRAW A LINE OR
LOOP: G50,G51      JUMP: G52,G53   SUBROUTINE: G54,G55
EXPRESSION: G59    ABS/INC: G90/G91
```

The user now knows the desired function is G01. He enters

```
N110   G01   X0     Y-6;    LINE A TO B
```

and the line segment AB is drawn on the output screen. The user then enters

```
N130   G10   C180;          ARC B TO C, CCW
```

to draw an arc. The system displays

```
**ENTER G12 TO DEFINE CENTER & RADIUS FOR ARC
```

The user now realizes that the center and the radius have to be defined before the arc can be drawn. If he needs further information about the function, he can enter the help mode by inputting CONTROL Z (see Appendix B.2). After the user inserts function G12, the program looks like this:

```
%;     PROGRAM START
N100   G92   X0     Y8;     START POINT A
N110   G01   X0     Y-6;    LINE A TO B
```

```
N120  G12  R2      C180;  DEFINE RADIUS & CENTER
N130  G10  C180;          ARC B TO C, CCW
```

Its output is shown in Figure 6. As soon as he sees it, the user realizes that the clockwise arc-drawing function has been erroneously used instead of the desired counterclockwise one. After requesting help messages, he replaces function G10 with G11, and the output changes as shown in Figure 7.

Now the user enters the statement

```
N140  G01  X0     Y6;     LINE C TO D
```

It executes without errors. The user then enters

```
N150  G01  X-4    Yo;     LINE D TO A
```

The CAPS displays the message

```
**ENTER A REAL NO.
(MAX. 9 DIGITS BEFORE AND 6 DIGITS AFTER ".")
```

and shifts to the editing mode, positioning the cursor at the character "o". The user can then change the letter "o" to the number "0".

Finally, the user enters the last statement:

```
N160  M02;                FINISH
```

The program is now correct and the output figure is drawn as shown in Figure 3.

Comparison of the CAPS and the CPS. Under the CPS, the user employed the editor and the translator six times and executed the program three times. The time spent on processing the editor and the interpreter, as well as the effort expended in shifting the system among the editing, translating, and executing modes, are the main disadvantages of the conventional programming environment. Referencing the manual further slows down the programming process and makes it even more cumbersome.

Under the CAPS, the user does not have to spend time shifting the system between different modes. It is automatically done by the CAPS. The guidance and help messages eliminate the inconvenience of referencing manuals. Thus, because the computer is friendlier, the programming job becomes easier and can be done faster.

Modeling programming systems with state diagrams

Plattner and Nievergelt pointed out that the state space concept can be used in the process of monitoring program execution.[16] This concept, combined with state diagrams (directed graphs),[17,18] can also be applied to the analysis and implementation of programming systems.

Modeling the CAPS with a state diagram. Figure 8 shows a simplified state diagram for the CAPS.[8] Each circle represents a system state. The label on each arc indicates the function which causes the state transition. The initial state, S99, is the state in which the program start block (%;) has not been entered or in which the end block (M02;) has been encountered; i.e., it is the state that exists before the program has been started or after it has been completed. The definitions of the states are given in Table 1 and the state flow table is shown in Table 2. Let us use the sample from the previous section to explain the state diagram.

At the beginning, the CAPS is in its initial state, S99. When the user enters

```
PROGRAM START
```

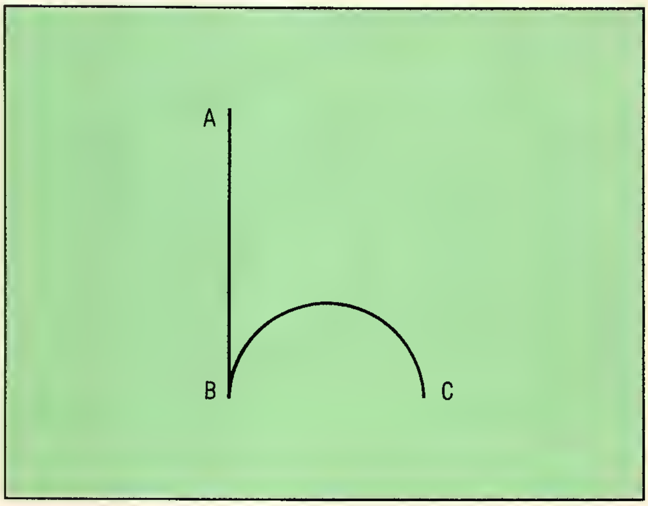


Figure 6. Output figure with semicircle facing in wrong direction.

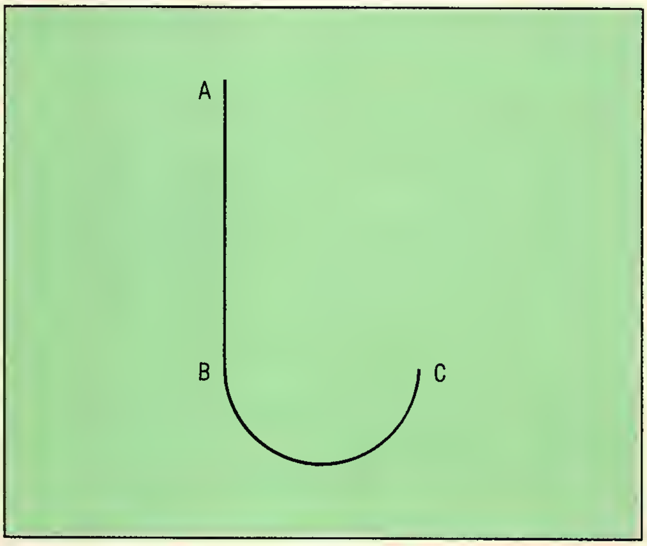


Figure 7. Output figure with semicircle facing in correct direction.

Table 1.
Definitions of CAPS states.

State	Description
S_{99}	Program not yet started (does not have %;) or already finished
S_0	Program just started (has %;)
S_1	Has first point for a line segment or an arc (has G92)
S_3	Has first point, center, and radius for an arc
S_7	Draw arc with first point, center, and radius defined

Table 2.
State table.

Next State		Input block						
		%	G92	G12	G10 G11	G01	M02	Other blocks
Present state	S_{99}	S_0	S_{99}	S_{99}	S_{99}	S_{99}	S_{99}	S_{99}
	S_0	S_0	S_1	S_0	S_0	S_0	S_{99}	S_0
	S_1	S_1	S_1	S_3	S_1	S_1	S_{99}	S_1
	S_3	S_3	S_3	S_3	S_7	S_3	S_{99}	S_3
	S_7	S_7	S_1	S_3	S_7	S_1	S_{99}	S_7

the system displays the guidance message and stays at state S_{99}. After the user changes the first statement to

```
%;      PROGRAM START
```

the system transfers to state S_0. The user then inputs the statement

```
N100   G92   X0    Y8;    START POINT A
```

The starting point of the line segment AB is drawn and the system changes to state S_1. When the user requests the help messages, the system enters the help mode, but this does not change the state of the system. After the user completes the statement

```
N110   G01   X0    Y-6;   LINE A TO B
```

the system is still in state S_1. The user now enters

```
N130   G10   C180;        ARC B TO C, CCW
```

The system does not go to state S_7. It just sends out error messages and remains in state S_1 due to the undefined radius and center of the arc. After the user inserts

```
N120   G12   R2    C180;  DEFINE RADIUS & CENTER
```

the system branches to state S_3. Then, the statement

```
N130   G10   C180;        ARC B TO C, CCW
```

is processed. A clockwise semicircle is drawn and the system goes to state S_7. After the user changes the function from G10 to G11, the clockwise semicircle is replaced with a counterclockwise one. But the system remains in state S_7. Now the user enters

```
N140   G01   X0    Y6;    LINE C TO D
```

The system outputs the line segment CD and branches back to state S_1. After the user inputs

```
N150   G01   X-4   Yo;    LINE D TO A
```

the CAPS displays error messages and remains in state S_1. The user now changes the letter "o" to the number "0" for the Y parameter. The line segment DA is drawn and the system branches to the same state S_1. When the user enters the end statement

```
N160   M02;               FINISH
```

the program is finished and the system goes back to the initial state, S_{99}. This state transition analysis can also be obtained by using the state table provided in Table 2.

State-diagram analysis of the CPS and CAPS. Examining the state diagram, we find that the reason the CPS cannot manage error recovery quickly and easily is because of the existence of the error states. Normally, a conventional programming system implicitly has many error states. Once an error is detected, the system transfers to one of the error states and the programming job is terminated. The user has to repeat all the procedures from the beginning in order to rerun his program; i.e., he has to start from the initial state again. He cannot return from the error state to the preceding state, correct the error, and continue the programming job.

In the CAPS, after the system detects an error, it automatically returns to the preceding state and waits for a correction. After the correction has been made, the statement is processed immediately and the system is transferred to its next designated state. The elimination of the error state and the automatic return to the preceding state are the key improvements which make speedy error recovery possible.

The CAPS dramatically reduces the programming effort in LNCM. The CAPS concept should be applicable to other high-level-language programming

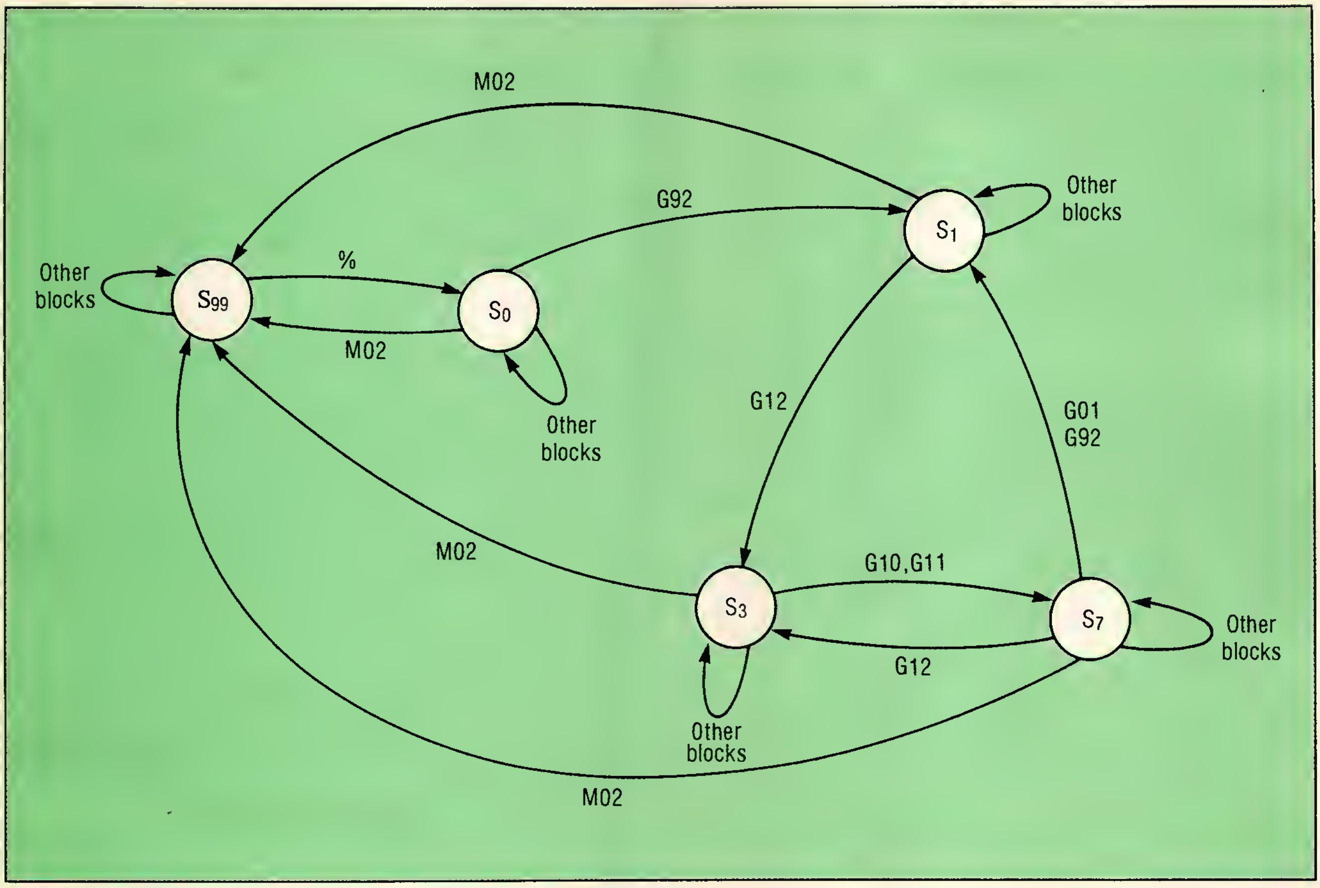


Figure 8. Simplified state diagram for the CAPS.

systems as well. State diagrams are very useful for analyzing such systems. However, the processing-time overhead introduced by the CAPS needs to be investigated.

In our application, the graphics output is used to simulate the operation of a numerically controlled machine tool. Such simulation should be useful in many other applications, where it can be employed to minimize material costs and time consumption. ■

Acknowledgment

The author wishes to express his appreciation to Erich Schmitt for his advice in, and discussion of, the design of the CAPS.

References

1. E. J. Lerner, "Software I1—Automating Programming," *IEEE Spectrum,* Vol. 19, No. 8, Aug. 1982, pp. 28-33.
2. E. J. Lerner, "Software 111—Programming for Nonprogrammers," *IEEE Spectrum,* Vol. 19, No. 8, Aug. 1982, pp. 34-38.
3. A. H. Singer, H. Ledgard, and J. F. Hueras, "The Annotated Assistant: A Step Towards Human Engineering," *IEEE Trans. Software Eng.,* Vol. SE-7, No. 4, July 1981, pp. 353-370.
4. L. Osterweil, "Software Environment Research: Directions for the Next Five Years," *Computer,* Vol. 14, No. 4, Apr. 1981, pp. 35-43.
5. A. K. Graham, "Software Design: Breaking the Bottleneck," *IEEE Spectrum,* Vol. 19, No. 3, Mar. 1982, pp. 43-50.
6. Y. Chu and E. R. Cannon, "Interactive High-level Language Direct-Execution Microprocessor System," *IEEE Trans. Software Eng.,* Vol. SE-2, No. 2, June 1976, pp. 126-134.
7. W. D. Hagamen et. al., "A Program Generator," *IBM System J.,* No. 2, 1975, pp. 102-124.
8. N. F. Yao, "Hardware and Software Design of A Friendly Computer Programming Environment: A Computer-Aided Programming System," PhD dissertation, Dept. of Elec. and Comp. Eng., State Univ. of New York at Buffalo, June 1982.
9. *The Am2900 Family Data Book,* Pub. no. AM-PUB003, Advanced Micro Devices, Sunnyvale, CA, 1979.
10. T. Teitelbaum and T. Reps, "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment," *Comm. ACM,* Vol. 24, No. 9, Sept. 1981, pp. 8-16.
11. O. Stromfors and L. Jonesjo, "The Implementation and Experiences of a Structure-Oriented Text Editor," *ACM Sigplan Notices,* June 1981, pp. 22-27.
12. C. W. Frasier, "Syntax-Directed Editing of General Data

Appendix A—Language for Numerically Controlled Machines (LNCM)

A.1 Language definitions

A modified BNF[19] is used to define the LNCM:

```
(1) Basic definition
    <digit>      ::= 0|1|2|3|4|5|6|7|8|9
    <letter>     ::= <address character>
                 ::= A|B|C|D|E|F|G|H|I|J|K|L|M
                     |N|O|P|Q|R|S|T|U|V|W|X|Y|Z
    <blanks>     ::= ƀ|<blanks>ƀ
(2) Arithmetic expression
    <primary>    :=  <register>|<unsigned decimal>
    <fun>        :=  &CO|&SI|&TN|&AC|&AS|&AT|
                     &SR
    <assignment> := [<letter>|<digit>]*
                      = <expression>
(3) Function definition
    <radius>     ::= R<decimal>
                     (A)
    <angle>      ::= {B}  <decimal>
                     (C)
(4) Program definition
    <program>    ::= <start block><block list>
                     <end block>
```

*Square brackets are used for both "option definition" and "register definition." Items enclosed in plain square brackets (with no *) are optional. The square brackets with * denote that the brackets are the symbols defined for the register.

A.2 Definitions of address characters[14]

ADDRESS CHARACTER	ADDRESS FOR
A	angular dimension around x axis
D	angular dimension around special axis, or third feed function, or tool function for selection of tool compensation*
F	feed function
G	preparatory function
H	permanently unassigned
I	interpolation parameter or thread lead parallel to x
N	sequence number or loop counter
P	third rapid traverse dimension or tertiary motion dimension parallel to x*
S	spindle speed function
T	tool function
U	second motion dimension parallel to x*
X	primary motion dimension

*Where D, P, and U are not used as indicated, they may be used elsewhere.

Structure," *ACM Sigplan Notices,* June 1981, pp. 17-21.

13. S. R. Wood, "Z—The 95% Program Editor," *ACM Sigplan Notices,* June 1981, pp. 1-7.
14. *TSO Full Screen Edit User's Guide,* Applied Software, Inc., Palm Beach Gardens, FL, 1980.
15. *EIA Standard RS-274D—Interchangeable Variable Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically Controlled Machines,* Engineering Dept., Electronic Industries Association, Washington, DC, Feb. 1979.
16. B. Plattner and J. Nievergelt, "Monitoring Program Execution: A Survey," *Computer,* Vol. 14, No. 11, Nov. 1981, pp. 76-93.
17. Z. Kohavi, *Switching and Finite Automata Theory,* 2nd ed., McGraw-Hill, New York, 1978.
18. D. Givone, *Introduction to Switching Circuit Theory,* McGraw-Hill, New York, 1970.
19. T. W. Pratt, *Programming Languages: Design and Implementation,* Prentice-Hall, Englewood Cliffs, NJ, 1975.

Neng Fred Yao is a senior engineer at Wang Laboratories in Lowell, Massachusetts. His main areas of interest include software engineering, microcomputer architecture, interactive graphics, and CAD/CAM systems. He received his BS degree in electrophysics from the National Chiao-Tung University, Taiwan, Republic of China, in 1973; he earned an MA degree in physics from the State University of New York at Buffalo in 1977, and MS and PhD degrees in electrical and computer engineering from the same institution in 1979 and 1982. He is a member of Tau Beta Pi and the IEEE Computer Society.

Questions about this article can be directed to Yao at Wang Laboratories, Inc., One Industrial Ave., Lowell, MA 01851.

A.3 Explanations of preparatory functions[14]

G CODE	FUNCTION	DESCRIPTION
G01	Linear interpolation	A mode of contouring control which uses information contained in a block to produce a straight line in which the vector velocity is held constant.
G10	Arc clockwise (point 2 CW arc)	An arc generated by means of polar coordinates in which the curvature of the path of the tool with respect to the workpiece is clockwise (when viewing the plane of motion in the negative direction from the perpendicular axis).
G11	Arc counterclockwise (point 2 CCW arc)	An arc generated by means of polar coordinates in which the curvature of the path of the tool with respect to the workpiece is counterclockwise (when viewing the plane of motion in the negative direction from the perpendicular axis).
G12	Arc start (point 1 of arc)	Specifies the radius and starting angle of the first point of an arc.

Appendix B—Guidance and help messages

B.1 Guidance messages

```
1. **ENTER PROGRAM START BLOCK "%;"
2. **ENTER G92: DEFINE START POINT   OR
           G12: DEFINE CENTER & RADIUS FOR ARC   OR
           G01: DRAW A LINE   OR
   LOOP: G50,G51        JUMP G52,G53    SUBROUTINE: G54,G55
   EXPRESSION: G59      ABS/INC: G90/G91
3. **ENTER G92 TO DEFINE START POINT BEFORE G12 OR G01
4. **ENTER G10 OR G11 TO DRAW AN ARC
5. **ENTER G92 & G12 TO DEFINE START POINT, CENTER & RADIUS OF ARC
6. **ENTER G12 TO DEFINE CENTER & RADIUS FOR ARC
```

B.2 Help messages

```
  HELP DIR : ENTER CONTROL Z,   EXIT HELP : ENTER CONTROL W
********** G(PREPARATORY) FUNCTION SUB-DIRECTORY **********
     LINE: G01,G92          ARC: G10,G11,G12
     LOOP: G50,G51          JUMP: G52,G53    SUBROUTINE: G54,G55
     EXPRESSION: G59        ABS/INC: G90/G91
       ENTER 2 DIGITS FOR DESIRED FUNCTION NUMBER.
```

This system employs low-cost electronic symbol scanners and a digital voice synthesizer to enable individuals who are both physically handicapped and mentally retarded to communicate more quickly and effectively.

Electronic Scanners with Speech Output—A Communication System for the Physically Handicapped and Mentally Retarded

Donald F. Hanson
University of Mississippi

Peggy Cook Power
North Mississippi Retardation Center

For many mentally retarded people, communication is very difficult. However, a symbol system can provide them with a means of communication. One such system—Blissymbols—enables the mentally retarded to communicate by means of graphic symbols. If, in addition to being retarded, an individual is also physically handicapped, as is often the case, then an aid to communication that takes into account the physical handicap is also required. This article describes the development of modularly compatible communication aids for physically handicapped, mentally retarded persons who use Blissymbols for communication at the North Mississippi Retardation Center (NMRC) in Oxford, Mississippi. The technical work was performed by senior engineering students in a design course at the University of Mississippi. (See "Project background," next page.)

The North Mississippi Retardation Center is a residential facility for mentally retarded persons which houses approximately 262 residents from the northern 23 counties of Mississippi. The staff of the center includes professionals from the fields of special education, psychology, nursing, physical education and recreation, and speech pathology, to name a few. NMRC is an educational facility and provides educational services to all residents in addition to other special services.

Institutionalized mentally retarded. Mentally retarded persons, who comprise approximately 3 percent of the total population of the United States, are categorized according to their intellectual and adaptive intelligence quotients into the following four areas:

(1) mild retardation (IQ of 52-67),
(2) moderate retardation (IQ of 36-51),
(3) severe retardation (IQ of 20-36), and
(4) profound retardation (IQ of less than 20).[1]

The etiologies of mental retardation are numerous and may include prenatal maternal diseases such as rubella, chromosomal disorders such as Down's syndrome, and

sociocultural causes such as poor nutrition. Hazards at birth, such as prematurity, anoxia, and injury to the head and brain (cerebral palsy), can also result in mental retardation.[2]

Among NMRC's mentally retarded residents there are a significant number of children and adults who are nonverbal. (A nonverbal, or nonspeaking, individual is generally defined as one who is severely communicatively handicapped because of limited or nonexistent speech abilities.[3]) The inability to speak may be a result of a neuromotor disorder such as cerebral palsy, in which case the individual may lack the motor coordination and integration needed to produce speech sounds. Deafness, autism (a disability resulting in disturbances of developmental rates and/or sequences, lack of response to sensory stimuli, and severe communication deficits[4]),

Project background

The multidisciplinary project which resulted in this article began in the summer of 1981 as an effort to provide volunteer student help to meet the communication needs of retarded and handicapped persons at the North Mississippi Retardation Center (NMRC) in Oxford, Mississippi. Since electrical engineering students at the University of Mississippi are required to design and build a senior project, it was decided that the best way the Electrical Engineering Department at the University of Mississippi could help NMRC was to arrange for interested seniors to design communication aids for the residents. The seniors received credit for EL.E. 462, "Senior Design II." "Senior Design II" and its predecessor, "Senior Design I," are requirements for the bachelor of science degree in electrical engineering at the University of Mississippi. Seniors are required to choose a design problem and go through the steps necessary to arrive at an original solution. This article describes the projects that were assigned to individual student volunteers or to student teams in the 1983 senior class, the second year of EE—NMRC cooperation.

Donald Hanson was the instructor for EL.E. 462 for both the 1982 and 1983 classes of EE seniors developing projects, and was the overall technical supervisor. Peggy Power was the overall supervisor expressing the needs of the NMRC population to the student workers. Karl Brenkert supervised the 1983 mechanical engineering seniors participating. The Oxford Kiwanis Club and the Telephone Pioneers, two local civic groups, provided funding for the second year (1983). Texas Instruments provided parts and equipment both years.

To understand the nature of the problem they were to solve, the engineering students visited NMRC to meet some residents with special communication needs and to learn about their motor skills and intellectual abilities. With their supervisors' guidance, they then looked at alternatives and started design.

The first year (1982). In the first year of the project, the seniors in EL.E. 462 were instructed to design a communication aid for a particular NMRC resident based on his or her motor abilities. This custom approach resulted in a short-term solution to the needs of NMRC; the devices worked for specific individuals and almost no one else. This approach was therefore abandoned the second year. A description of the 1982 projects can be found in Fabian.[1]

Also during the first year, a request was made to Texas Instruments for donations of speech integrated circuits and systems. They donated a prototype of their "speech education module" and agreed to program the NMRC word list given in Appendix A for 5220 compatibility.

The second year (1983). The experience gained the first year was useful to the final success of the 1983 projects because the needs of NMRC residents were better understood. In 1983 the students were required to design systems which were modularly compatible. The control switch and an EPROM containing the word codes (Appendix A) for a person's communication device were the only items customized to individual NMRC residents. By changing the control and the EPROM, any scanner could operate with any control. Each scanner could operate in either a continuous-scan mode or an *x-y* mode. The speech synthesizer was designed to function properly with any control/scanner/EPROM combination. The fruit of this approach is described in this article. The donated TI hardware and EPROMs arrived in time for use in the second year. Local support from civic groups provided the necessary funds for other parts and equipment.

On the basis of the needs of NMRC's population, individual students or groups of students were assigned to design the control/scanner interface (Farris), the scanner/voice synthesizer interface (Smith), the voice synthesizer (Coleman, Lantz, and Smith), the 4 × 4 scanner (McNeer), the 5 × 5 scanner (Logan, Brenkert, and White), and the 8 × 8 scanner (Ronaldi, Tan, Anderson, and Baker). The motivation level of these students was very high because they were excited about helping people. Since the senior year ends in graduation, those parts of the projects that needed to be finished or refined after graduation were revised by electrical engineering supervisor Donald Hanson. This amounted to adding EPROMs to the 4 × 4 and

dysarthria (speech disorders produced by peripheral or central nerve damage[5]), and severe or profound mental retardation are other factors which prevent the adequate development of functional speech.

It is often difficult for people who work with nonverbal individuals to estimate how much they are able to understand, or the extent of their communication and intellectual abilities. In addition, individuals who lack the ability to communicate verbally and who have no alternative communication system because of physical handicaps often become frustrated and depressed at their failure to communicate. The inability to communicate makes it very difficult for these persons to manipulate their environments, and they may eventually become passive and dependent on others for the fulfillment of their needs. The nonverbal population presents a great need for a form of augmentative, or alternative, communication system to supplement their inadequate speech skills. An autonomous communication aid can greatly increase the independence, socialization skills, and learning abilities of many nonverbal individuals. In some cases, the development of a communication system has the effect of increasing an individual's measurable intelligence.

Blissymbolics. One form of alternative communication which has been found to work very well with many mentally retarded individuals is Blissymbolics. Blissymbolics is a form of graphic communication in which words, concepts, and ideas are represented through symbols.[6] Many Blissymbols are *pictographic* in that they resemble the object they represent, such as the symbol for "chair" in Figure 1a. Others are *ideographic* and depict the idea of something, such as the symbols for "in" and "out," shown in Figures 1b and 1c, respectively. *Arbitrary* symbols represent those things for which there is no concrete object, such as the Blissymbol for "the" shown in Figure 1d. A fourth category of Blissymbols encompasses the *international symbols,* such as numbers and the question mark, which represent basically the same idea in most cultures of the world. Blissymbols can be combined to form compound symbols when necessary; e.g., the symbol for "desk" in Figure 1e is a combination of the pictographic symbol for "table" and the symbol for "to work"—a desk is a table at which you work. The system allows for the production of full grammatical sentences; by pointing to each symbol in turn, the nonverbal individual can produce a sentence such as "I sit at the desk." The words are always included either above or below the symbol to make it an open system, allowing someone who does not recognize the graphic symbols to communciate with a Blissymbol user. Although you, as the reader, may comprehend Blissymbols more easily by analyzing the component parts of a symbol to derive its meaning, the mentally retarded person learns the symbol holistically, recognizing the entire symbol as representing one meaning.

Blissymbolics was invented by Charles Bliss, an Austrian-born chemical engineer who spent part of the Second World War in a German concentration camp.[7] Bliss believed that many of the world's problems were a result of a lack of communication between different-speaking persons. He spent much of his life working at an international language which could be used by all people and published his book *Semantography* in 1949.

8 × 8 scanners and simplifying the speech synthesizer software (Appendix B). Everything else was designed by the students with guidance from supervisors when requested. It is interesting that all three groups working on scanners used slightly different principles for scanning.

The impact this project has had on the students and on the community has been significant. NMRC has obtained communication aids for their residents at no cost; the students have been exposed to new concepts through a multidisciplinary humanitarian project; and lines of communication between the electrical engineering, mechanical engineering, communicative disorders, special education, and physical therapy fields have been developed.

Many people contributed to this project during 1982 and 1983. Contacts at Texas Instruments were Dave Allen, Gene Bluhm, Reed Borie, Mike Hutchins, Brian Joslin, Terryl Joslin, Bernie List, Tony Monterosso, and John Ribe. Dr. Charles E. Smith, Sr., originally suggested this joint effort. Electrical engineering students from the class of 1982 who designed and built projects were Mohamed Al-Fakhouri, Dimitri Diliani, Brian Fox, Rusty Morphew, and Jeff Zysk. Class of 1983 electrical engineering students involved in the project were Stephen R. Coleman, Gerald B. Lantz, Robert L. Logan, Jr., George R. McNeer, Thomas C. Ronaldi, Charles E. Smith, Jr., and Boon P. Tan. For 1983, mechanical engineering students under the direction of Dr. Karl Brenkert were John Anderson, Steve Baker, Eric Brenkert, and Howard White. Others who participated were Dr. Mark Tew, Tim Farris, Tracy Baker, Ralph Taylor, Demos Demosthenes, and Steve Scott. Unless otherwise noted, the photographs were taken by Bill Martin. Ray Cronin and Kenny Pruett made the printed-circuit boards. Dr. George Duncan and Dr. Martin Rothenberg of Syracuse University contributed to the sidebar on 5220 operation.

Reference

1. D. K. Fabian, "Communication Is Bliss," *The Ole Miss Alumni Review,* Vol. 31, No. 2, June 1982, pp. 14-16.

However, linguists showed little acceptance of his new communication system.

In 1971, a teacher at the Ontario Crippled Children's Centre, Shirley McNaughton, came across Bliss' system while looking for a communication system she could use with her cerebral-palsied, nonambulatory students.[8] The results were remarkable. Children who had never been able to communicate their thoughts or ideas before could now do so. The use of Blissymbolics became widespread, extending to deaf, autistic, mentally retarded, and stroke-victim populations. In 1975, the Blissymbolics Communication Institute was established to standardize the use of Blissymbolics throughout the world.[9]

Blissymbols at NMRC. Use of Blissymbols at NMRC began in 1980. NMRC's first Blissymbol user was able to learn the system quickly, and more children and adults were enrolled in the program. Today, the center has over 20 residents who use this system of communication. They range from those with three-word Blissymbol vocabularies to the center's first Blissymbol user, who has a vocabulary of approximately 250 symbols. NMRC's ambulatory residents often carry portable trifold boards containing the symbols so that their system of communication is always at hand. For the nonambulatory, cerebral-palsied Blissymbol user, a lapboard attached to the wheelchair provides easy access to the symbol system.

A more automated communication aid containing Blissymbols is often desirable for those residents who, because of severe motor handicaps, have difficulty pointing to the symbols on wheelchair lapboards. Since electronic devices can greatly decrease the time needed by a severely motor-impaired person to transmit a message, their use can result in an accompanying decrease in communicative frustration and increase in expressive skills.

Communication aids. Communication aids, which may be defined as physical mechanisms combined with symbol systems, have undergone many modifications over the past few decades, with the major changes involving the use of microcomputers to produce more complex and comprehensive aids. Aids range from simple, handmade, custom-designed picture and word boards to portable electronic aids with LED displays, television displays, strip printers, and typewriter controls.[10] Many communication aids are also now designed to provide input to a computer, enabling the handicapped individual to tap many previously unavailable resources. The complexity of the aids is variable, and the price of the communication devices reflects this complexity.

Fundamental communication aids usually do not incorporate moving parts or electronic components, and can be constructed by laymen or teachers. This category includes alphabet, word, or Blissymbol boards through which the individual can communicate by pointing or indicating a letter, word, symbol, or general area of the board—a process known as direct selection. Encoding methods, which require fewer movements and allow for an increased lexicon, can be used successfully with fundamental aids. Manual scanning aids, in which the elements are presented one at a time to the handicapped individual, who then indicates by a prearranged method (such as a head movement or eye blink) when the desired element has been pointed to, are used effectively with severely physically impaired persons.[11]

Simple electronic aids are slightly more complex than fundamental aids, and require electronic or mechanical components for operation. With an electronic aid, the need for the message recipient to actively or manually assist in the communication process is great diminished. The handicapped individual has more freedom and versatility since he or she can allow the aid itself to scan lexical items, most often using LEDs. Such an aid is often referred to as a scanner. Three Blissboard scanner circuits are detailed in this article.

Fully independent aids are those which contain a printout or display. With this type of augmentative communication device, the nonverbal individual can record and store messages, further releasing the message recipient from the need to give constant one-on-one attention.

Fully independent portable aids differ from the above only in that the printout does not require a typewriter or other piece of stationary equipment. This allows the individual to move about during communication.

Although augmentative communication aid technology has progressed in the last two decades, there still remain many drawbacks to the current resources in this area. Many electronic and independent aids are prohibitively expensive, ranging up to the $7000 mark.[12] While communication devices suited to the cognitive and communicative abilities of NMRC residents are not this expensive, aids that would allow for growth of the student's vocabulary, and possibly his motor abilities as well, would, in most instances, cost thousands of dollars each. This high cost would make it difficult, if not impossible,

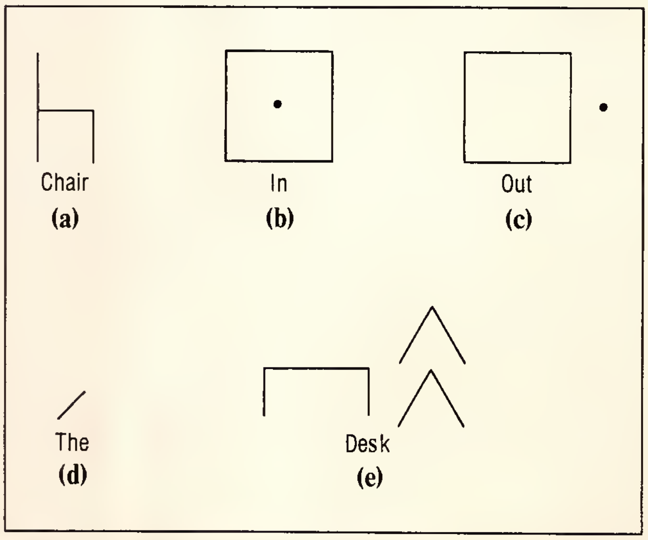


Figure 1. Examples of Blissymbols.

for families of NMRC residents, or the facility itself, to purchase them. A large part of this cost is due to the time and money spent by the companies in developing the devices and demonstrating their use. Although the cost of expressive communication devices should decrease in the future, the market for these aids will never be as great as the market for calculators, which underwent a similar development process.

Special needs of the handicapped and retarded. The population which could benefit from the devices presented in this article consists of those persons who are both physically handicapped and mentally retarded. Since many retarded individuals cannot learn the complex processes needed to acquire reading skills, words or the alphabet cannot be used as a viable means of communicating with them. A symbol system, such as Blissymbols, is required instead.

Within the physically handicapped population, each individual has unique motor abilities that he or she can use to control an electronic communication aid. Therefore, controls for communication aids must be customized to individual needs. When a custom device is developed to solve a particular individual's communication problem, it is much more effective if it exploits the person's existing motor and communication abilities.

NMRC's retarded Blissymbol-using population cannot read or talk effectively, and each individual has special motor abilities which must be considered in the design of an electronic communication aid. For example, one of the center's Blissymbol users, a severely impaired cerebral-palsied male, has no functional control of his hands or head. He can control the movement only of his eyes and tongue. At the present time, this individual uses a scanning approach—he looks at a specific area of the Blissboard while the person with whom he is communicating points to each Blissymbol in turn, waiting for a "yes" response (a tongue movement). This is a time-consuming and laborious process for all involved, including the Blissymbol user. This is but one example of the population for whom communication devices were developed. In general terms, the population shares the following traits:

- they are both mentally retarded and physically handicapped;
- they can hear and understand speech;
- they have control of an arm, leg, foot, or the tongue;
- they cannot read but can identify Blissymbols; and
- they cannot speak intelligibly.

Modularly compatible communication aids. The engineering problem was to design and build communication aids that meet the needs of this special population. Since the population is retarded and cannot read, but can identify Blissymbols, Blissymbols were used for communication of ideas. Since the population is also physically handicapped and can understand speech, but cannot speak, a simple electronic aid or scanner with speech-synthesizer capability was chosen. All systems were required to be modularly compatible to avoid duplication of effort. To meet the need for customizable controls, each scanner was designed to operate with a wide variety of input devices and to function in two modes, the continuous scan mode and the *x-y* mode. Hence, an individual can operate different Blissymbol scanners simply by switching custom controls. Each of the Blissboards was designed to operate with the speech synthesizer. Since low cost was a primary objective, none of the Blissymbol scanners themselves use a microprocessor, but the speech synthesizer does. Finally, because of the desire for portability, the scanners were designed to operate with batteries when not connected to the speech synthesizer.

Figure 2 shows a block diagram of a typical system. The control of the entire system is provided by a control switch subsystem customized to each individual's physical handicaps. The control leads to the scanner itself. The scanner scans while the input control signal is active, and a selected Blissymbol is indicated by a lit LED or lamp

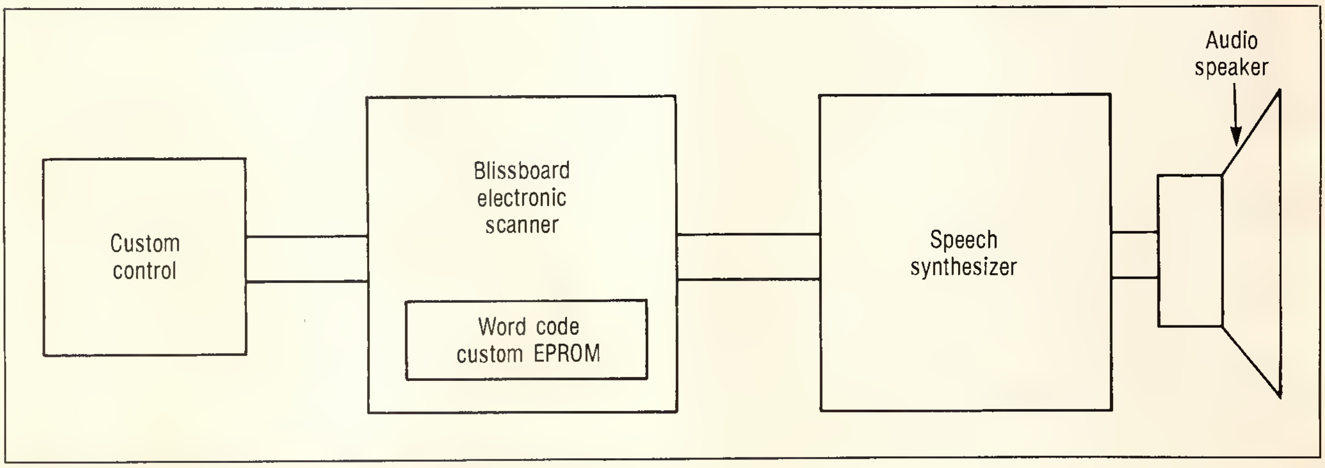


Figure 2. Typical electronic Blissymbol communication aid.

while the input control is inactive. Each Blissboard includes an EPROM containing selection addresses for the speech synthesizer so that the speech synthesizer knows which Blissymbol has been selected. This EPROM is the only part of each scanner that has to be customized. The voice synthesizer receives the selection address from the Blissboard EPROM and speaks the addressed word on command (when a SPEAK line is asserted).

The speech synthesizer is one feature of the system that is particularly relevant to Blissymbol training. The synthesizer adds a self-monitoring capability to the Blissboard scanners. When an individual selects a particular symbol, causing the word to be spoken, he automatically knows if his selection was correct. This enables a child to learn at his own rate and does require that a teacher or speech pathologist be with him constantly to supervise, train, and correct him. The child receives immediate auditory feedback as to the correctness and communicative intent of his message, and also perceives how an expressive communication system can assist him in communicating with others. Texas Instruments is now distributing a new device, the Vocaid, which is a word-and-phrase communication board equipped with synthetic speech output. The Vocaid fulfills a great need in the field of communication devices, and it can be adapted to use with Blissymbols. However, the Vocaid seems best suited to individuals who have fairly good manual dexterity and average-to-high cognitive abilities.[13]

Design of the electronic Blissymbol scanner system

The scanner systems consist of controls, the Blissymbol scanners, and the speech synthesizer. Designs for three controls, three scanners, and a speech synthesizer are included here—4×4, 5×5, and 8×8 scanners were all completed. Low cost was the primary consideration in all designs. The systems were designed to operate on batteries (when the speech synthesizer is not attached) for portability. Two of the systems use CMOS so that the circuitry can be enclosed in a box for safety.

Controls. Since the devices are to be used primarily by those unable to point or reach, remote switches were chosen for control. The control of the system can be easily customized to an individual's particular needs. Figure 3 shows three examples of input controls—a tongue switch,

Figure 3. Typical controls—a tongue switch (left), an *x-y* button panel (top right), and a joystick (bottom right).

an x-y button panel, and a joystick. Each of these controls may be used interchangeably among Blissboards. A nine-contact connector is used to attach each of these controls to the Blissboard. The pin assignments for this connector are shown in Table 1. Figure 4a shows the schematic of an x-y-mode joystick, and Figure 4b gives the schematic for a continuous-mode switch.

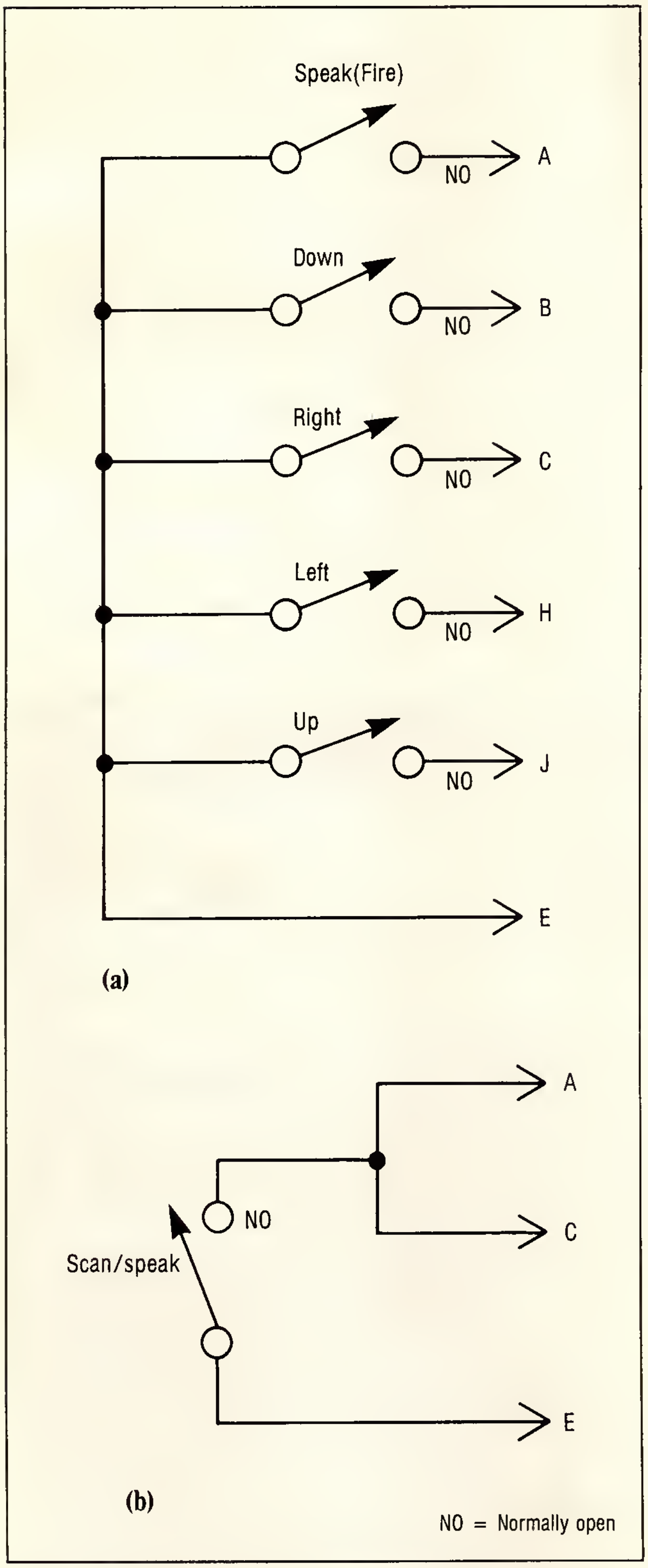


Figure 4. Control schematics—(a) for a typical x-y control (here, a joystick); (b) for a typical continuous-mode control (here, a tongue switch).

The tongue switch operates a Blissymbol scanner in the continuous mode. When the tongue is held up, the scanner actively scans (Pin C), and when the tongue is relaxed, the switch opens, scanning stops on a particular Blissymbol, and the word associated with the Blissymbol is spoken (Pin A) through an audio speaker. This type of control is particularly useful for smaller boards. For a large board, selection time may be long. However, the continuous mode works quite effectively with a 4×4 or 5×5 scanner.

The other two controls in Figure 3 operate a Blissboard in an x-y fashion, and a switch closure results in motion in the desired direction—UP, DOWN, RIGHT, or LEFT. diagonal movement is also possible. The fire button on the joystick serves as a SPEAK button. The selected word is spoken when the fire button is released. The x-y button panel has the same schematic as the joystick and can be used by those children physically unable to manipulate a joystick successfully.

Electronic design of the 4×4 scanner. A photograph of the 4×4 scanner is shown in Figure 5. The control enters on the left and the speech synthesizer is attached through the connector on the right. This particular unit was designed for a young child with cerebral palsy.

The scanner uses two bits each of two four-bit binary up/down counters when operating in the x-y mode. The schematic is given in Figure 6. One counter (y) is associated with the vertical scans and the other (x) handles the horizontal scans. The value of the y counter is fed to a 2×4 decoder which enables the row of the selected LED (Blissymbol), and the x counter is fed to another 2×4 decoder which selects the column of the selected Blissymbol. When the UP or DOWN control is asserted, the y counter is actively counting. When the LEFT or RIGHT control is asserted, the x counter is actively counting. Figure 7 shows the digital address of each LED in the array. From this assignment, one can see that to move down, the y counter must count up, and to move up, the y counter must count down. When the joystick is in a neutral position, all 4490 outputs are high. Thus, when UP is activated, this line goes low, telling the y counter to count down.

For continuous operation, only one four-bit counter is required, as can be seen from Figure 7. Therefore, when the mode switch is set for continuous operation, the two least significant bits of the x counter go to the x decoder, while the two most significant bits of the x counter are routed to the y decoder through a 4019 multiplexer, as shown. In this mode all LEDs are cycled through one at a time as described previously.

The address from Figure 7 which goes to the x and y decoders is also sent to a 2716 EPROM. The word at y,

x = 0000 in the top left hand corner is "yes." For the speech synthesizer to say "yes," it must receive hexadecimal (signified by \$) 4A as its input address. Therefore, EPROM address zero contains \$4A. Such programming allows each Blissboard to say the proper words. Only an EPROM needs to be programmed to make a change. The word list and associated addresses are given in Appendix A. When the SPEAK button is released, the 4047 one-shot puts out a negative pulse. The negative edge causes the speech synthesizer to say the selected word.

Electronic design of the 5 × 5 scanner. A photograph of the 5 × 5 scanner is shown in Figure 8. It will be the basis of a 100-symbol device having four "screens," each of 25 symbols, on a length of plastic film. In the continuous mode, when one screen is scanned, a motor will move the plastic film so that the next screen appears on the scanner. The motor control system needs to be designed before this project is complete.

The circuit diagram of the 5 × 5 array is given in Figure 9. Binary up/down counters are used here, too. This time, however, the design is not as straightforward as that of the 4 × 4 array. This is because 5 is not a power of 2, as was 4.

In the *x-y* mode, operation is much the same as for the 4 × 4 scanner except that both the asynchronous parallel load (preset) and the asynchronous clear (reset) are used. The counters must be immediately reset to zero when they reach five, so for practical purposes the count goes 0, 1, 2, 3, 4, 0, 1, ... This is accomplished by making reset $R = Q1 \cdot \overline{Q2} \cdot Q3$, as shown in Figure 9. When counting down, on the other hand, a counter must be set to four immediately when 1111 is reached. For practical purposes, the count then goes 4, 3, 2, 1, 0, 4, 3, ... This is accomplished by making $PE = Q2 \cdot Q3$, as shown in Figure 9. The outputs of these counters are fed to the x and y decoders, as shown. In the continuous mode, the y counter is incremented by means of a carry-in from Q3 of the x counter. With only a few minor changes, this

Table 1.
Control connections.

Connector pin	Blissboard function
A	Speak
B	Down
C	Right/continuous
D	Future expansion
E	GND
F	Future expansion
H	Left/continuous
J	Up
K	Future expansion

Figure 5. The 4 × 4 scanner.

circuit can be used to control an $m \times n$ array, $m,n \leqslant 16$. Control of the scan rate is provided by a potentiometer on the timer.

Electronic design of the 8 × 8 scanner. A photograph of the 8 × 8 scanner is shown in Figure 10. This scanner is intended for classroom use and so is not portable. Light bulbs are used instead of LEDs. This design is a form of accumulator; its circuit diagram is shown in Figure 11. When the joystick is not active, zero is added to the present accumulated value, which is retained by six D flip-flops. The least significant three bits go to the 3 × 8 decoder (x) which controls the column of the selected Blissymbol, and the most significant three bits go to the 3 × 8 decoder (y) which selects the active row. The D flip-flop outputs also go to a 2716 EPROM containing the speech addresses. To move to the right, one (1) must be added to the present value of the accumulator. To move left, minus one—or 111111—must be added. To move up, minus 8 or 111000 must be added, and to move down, plus 8 or 001000 must be added. These numbers result from using two's-complement numbers. The OR gates shown at the A inputs to the adder in Figure 11 are used to generate the proper numbers, as described above.

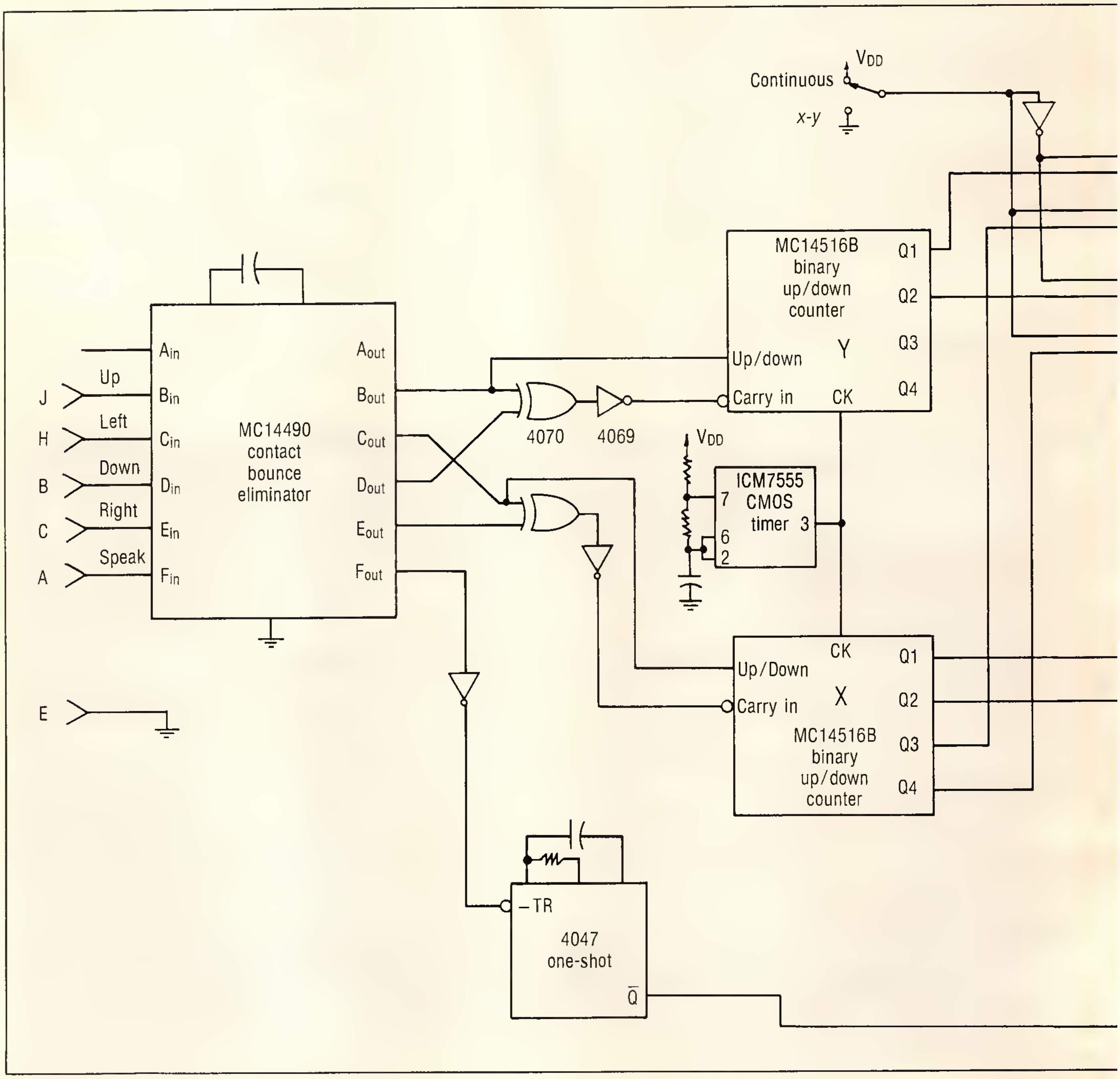
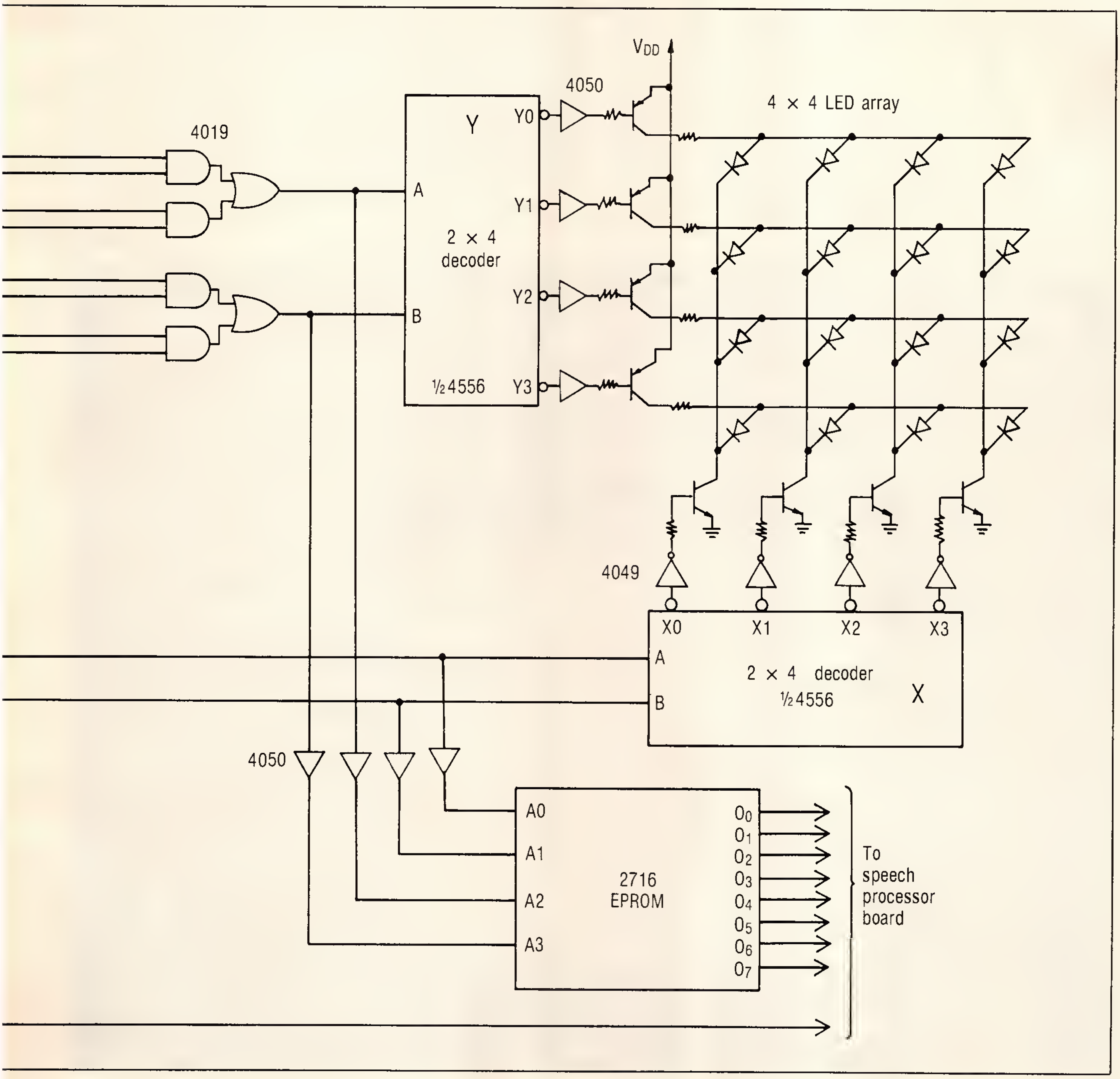


Figure 6. Simplified circuit diagram for the 4 × 4 array (routine connections omitted).

One advantage to this system is that no mode switch is required. Operation is in both modes at all times.

The speech synthesizer

The addition of a speech synthesizer to this system was made possible by a donation from Texas Instruments, who processed 190 words and phrases for NMRC so that its residents could share a common vocabulary. There are several reasons for selecting the Texas Instruments TMS 5220 Voice Synthesis Processor (VSP) over all others. First, since the words need to be said one at a time (out of context), speech quality must be excellent. It is well known that high-quality speech is produced with TI's linear predictive coding (LPC) with a minimal number of bytes of storage. Therefore, this lowers cost because fewer EPROMs are required. Lastly, since the vocabulary required by NMRC's Blissymbol users is very small (between 3 and 250 words), systems based on phonemes or allophones for text-to-speech conversion are unnecessary. Thus, a vocabulary of 190 words provides more capability than necessary for most of the individuals who are to be aided by the system.



0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Figure 7. Addresses of Blissymbol locations.

A photograph of the speech processor is shown in Figure 12. The WORD CODE and SPEAK lines labeled "to speech processor board" in Figures 6 and 11 are routed to the speech processor board through the dual-in-line cable connector shown in the photograph. In this prototype, a motherboard-daughterboard arrangement is used, among other reasons, to permit board swapping for ease in troubleshooting. The system has operated reliably to date.

The schematic diagram of the speech synthesizer is shown in Figure 13. The inputs from the Blissboard are an 8-bit WORD CODE and a SPEAK line. When SPEAK is low, data from the Blissboard's EPROM are latched into eight D flip-flops with the $\phi 2$ clock. The negative edge on SPEAK sets a bit in the 6520 PIA. The setting of this bit by the SPEAK input initiates the synthesis of the word whose WORD CODE is present on the data lines input from the Blissboard. At the top left of the figure are the seven 2532 4K $\times$ 8 EPROMs provided in the donation by Texas Instruments. The 3 $\times$ 8 decoder shown immediately to their right provides the chip selects for these seven chips. The top 2532 is selected by any ad-

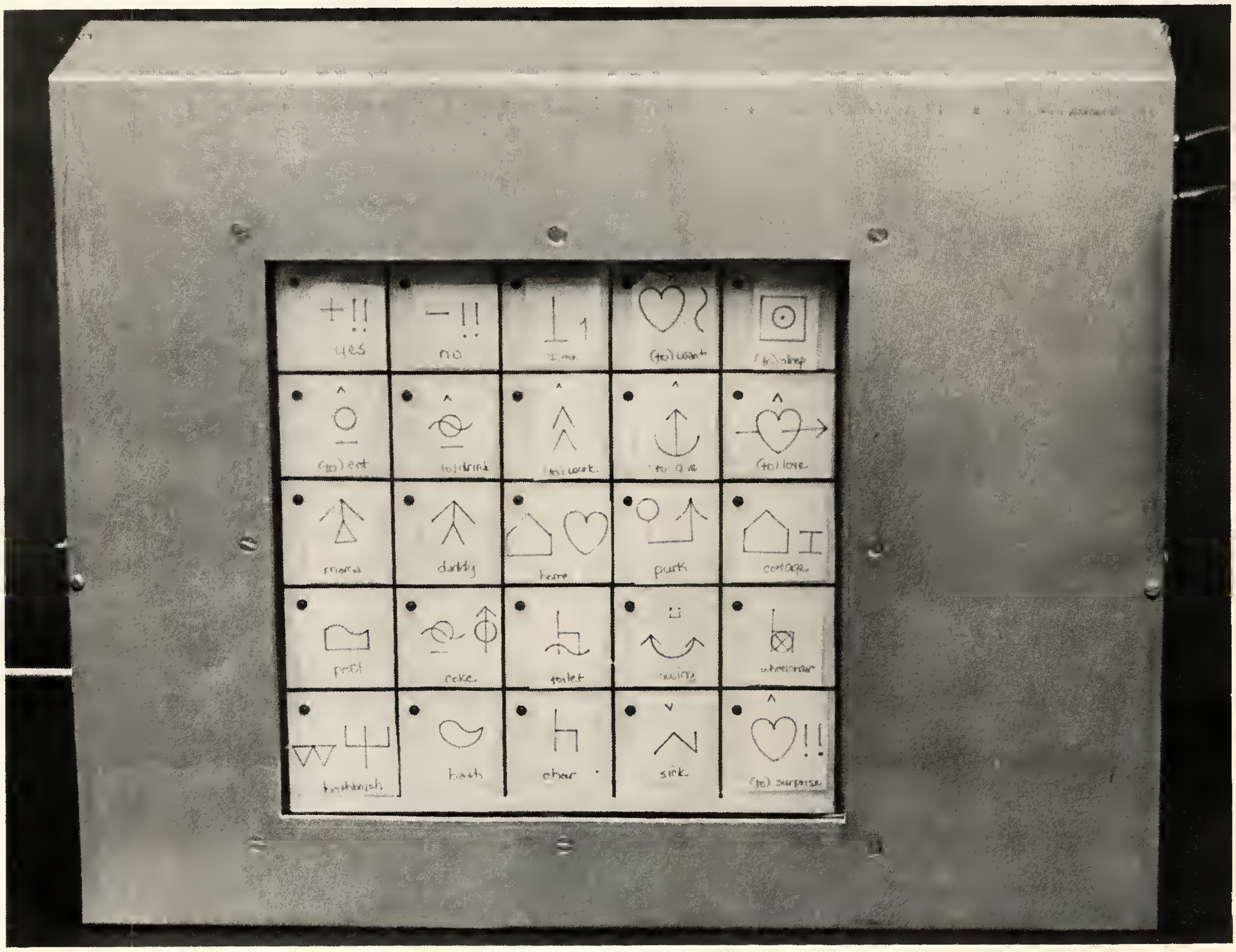


Figure 8. The 5 $\times$ 5 scanner.

dress with a hexadecimal 1 in the most significant nibble. A $2 selects the next 2532 and so on through chip 7. The 190 words and phrases are stored in LPC form in these EPROMs. In order for a word to be spoken, the starting address of the LPC data for that word must be looked up, and then the *n* bytes of LPC data beginning at that address, followed by one byte of $00,[14,15] must be routed to the speech processor through the 6520 PIA. For simplicity, the starting address and number of bytes in the word are stored in a 2716 mapper EPROM. This EPROM is selected by any address with a most significant nibble of $8. Note, however, that only A0 and A1 of the mapper EPROM can be controlled by the 6502 microprocessor. The eight address lines A2-A9 are controlled by the WORD CODE input from the Blissboard. Therefore, for every WORD CODE input from the Blissboard, four memory locations can be accessed in the mapper EPROM by the 6502. These four locations contain the starting address and number of bytes of the encoded word, as shown in Table 2. This scheme eliminates the need for the speech processor to know what the WORD CODE is. Instead, the starting address and number of bytes are available directly in location $8000-$8003. Besides those components already described, Figure 13 shows 2114 RAMs (pages $02 and $03 not used), a reset circuit, the 6502 microprocessor, and the 2716 program EPROM, which is at location $F800. At right, the audio speaker is driven by an LM383 audio amplifier.

Word list. The word list was selected on the basis of past experience at NMRC in using Blissymbols for communication, education, and daily living needs. Appendix A gives the WORD CODES for the NMRC speech synthesizer along with the starting address and number of bytes for each word. It is the latter data that are stored in the mapper EPROM. Some of the words listed in Appendix A have relevance to NMRC residents but may not have meaning to those not familiar with the institution. First, "cottage" is where a resident lives at NMRC, whereas "home" is the home of the resident's parent or guardian. NMRC residents work on a "token" system and instead of receiving money, receive tokens for good behavior or for completing tasks. These tokens are redeemable at the NMRC "token store" for toys and clothing. The "canteen" is a snack bar.

TMS 5220 Voice Synthesis Processor. A simplified description of the operation of the TMS 5220 VSP appears in "TMS 5220 operation," at right. Here, we will deal with the interfacing and control requirements for our application. The 5220 VSP is controlled by the 6502 microprocessor through a 6520 PIA. Pull-ups are required for interfacing to the 6520 PIA, as shown in Figure 13. The data manual[15] states that 5220 VSP power-up should be completed in 2 ms or less. Therefore, the 5220 is powered up with PA0 (6520), as shown in the sche-

TMS 5220 operation

The TMS 5220 Voice Synthesizer Processor (VSP) is a digital equivalent of a human vocal tract. It is based on a model of one individual's vocal tract, although good results can be obtained for a variety of speakers. Two types of speech sounds can be synthesized with the VSP, namely, voiced and unvoiced. The vocal excitation source for voiced sounds is a quasiperiodic vibration of the vocal cords, which produces a broad-spectrum, pulse-like, quasiperiodic "source" waveform at some fundamental frequency or *pitch*. Unvoiced sounds are made by a speaker by shaping the vocal tract so as to produce a source of aperiodic turbulence at some point, but not necessarily at the vocal cords. Both types of sounds can be voluntarily produced with a certain intensity or *energy* level. The voiced or unvoiced source wave propagates through the vocal tract and is changed or *filtered* by the shape of the oral cavity and the position of the lips and tongue. The result is that a part of a word is said over a time Δt that is small enough so that the configuration of the vocal tract is essentially constant. To say a complete word of duration t, the number of word parts that need to be said is, then, $n = t/\Delta t$. A natural perceptual interpolation by the listener occurs from one word segment to the next because of the slow rate of change of articulatory posture in the speaker. The 5220 VSP uses these concepts to reproduce a speaker's patterns of source and vocal-tract change for a vocabulary of words and phrases. By exploiting the natural interpolation performed as a result of the limitations on vocal-tract dynamics, the 5220 VSP can simulate the human vocal tract by concatenating sounds and thereby can produce synthetic speech. As we shall see shortly, then, the key parameters needed by the 5220 are FILTER parameters, PITCH, and ENERGY.

The data that are fed into the 5220 VSP are preprocessed by a speech development system.[1] The input to this system is a tape recording of the word spoken by a speaker. The development system breaks the word up into analysis frames of length $\Delta t = 25$ ms, and returns a binary number of up to 50 bits in length for each complete frame of speech. The binary data returned for each 50-bit frame of speech include 4 bits for ENERGY, 1 bit for REPEAT, 6 bits for PITCH, 5 bits each for K1-K2, 4 bits each for K3-K7, and 3 bits each for K8-K10. The 10 parameters (K1,K2, . . . ,K10, termed REFLECTION coefficients) are coded filter parameters associated with the size and shape of the oral cavity as modeled by the development system. The REPEAT bit indicates that the last frame's filter parameters K1-K10 are to be repeated in the present frame. An ENERGY = 0000 results in silence; ENERGY = 1111 is used to indicate it is time to stop talking; otherwise, ENERGY is coded to represent the vocal intensity. PITCH = 000000

is used for unvoiced speech, for which only reflection coefficients K1-K4 are needed. If the PITCH parameter Is set to some nonzero value, on the other hand, then the data refer to voiced speech, for which all ten reflection coefficients K1-K10 are required; in this case PITCH is related to the fundamental frequency of the excitation source. The number of bits generated by the development system for each of the above cases is 4 bits for no speech (silence or stop), 11 bits for "repeat last reflectlon coefficients," 29 bits for unvoiced speech, and 50 bits for voiced speech.

The heart of the 5220 is a PMOS ten-pole digital lattice filter whose filtering parameters are derived from the ten 5-, 4-, or 3-bit K parameters which were initially generated by the development system. Fourteen-bit arithmetic is used internally for speech calculations. The digital filter's REFLECTION coefficients and the PITCH and ENERGY coefficients are updated eight times per 25-ms frame by a special frame-to-frame interpolation scheme. For voiced speech, a periodic impulse is injected into the filter system. The impulse height is determined by a ten-bit gain parameter derived from the 4-bit ENERGY field, and the periodicity of the impulse input is determined from a ten-bit parameter derived from the 6-bit PITCH field. When unvoiced speech is selected (PITCH = 000000), the input to the digital filter is a random train of impulses, or white noise, whose power is determined by a ten-bit gain parameter derived from the 4-bit ENERGY field.

The lattice filter, as defined by the set of reflection coefficients (K1-K4, unvoiced; K1-K10, voiced), serves to modify the spectral characteristics of the filter excitation input in the same way that the vocal tract configuration filters the vocal excitation function. The output signal from the lattice filter is 14 bits wide and represents the synthesized speech, whose time- and frequency-domain characteristics ideally closely resemble the original speech signal encoded by the development system. The excita-

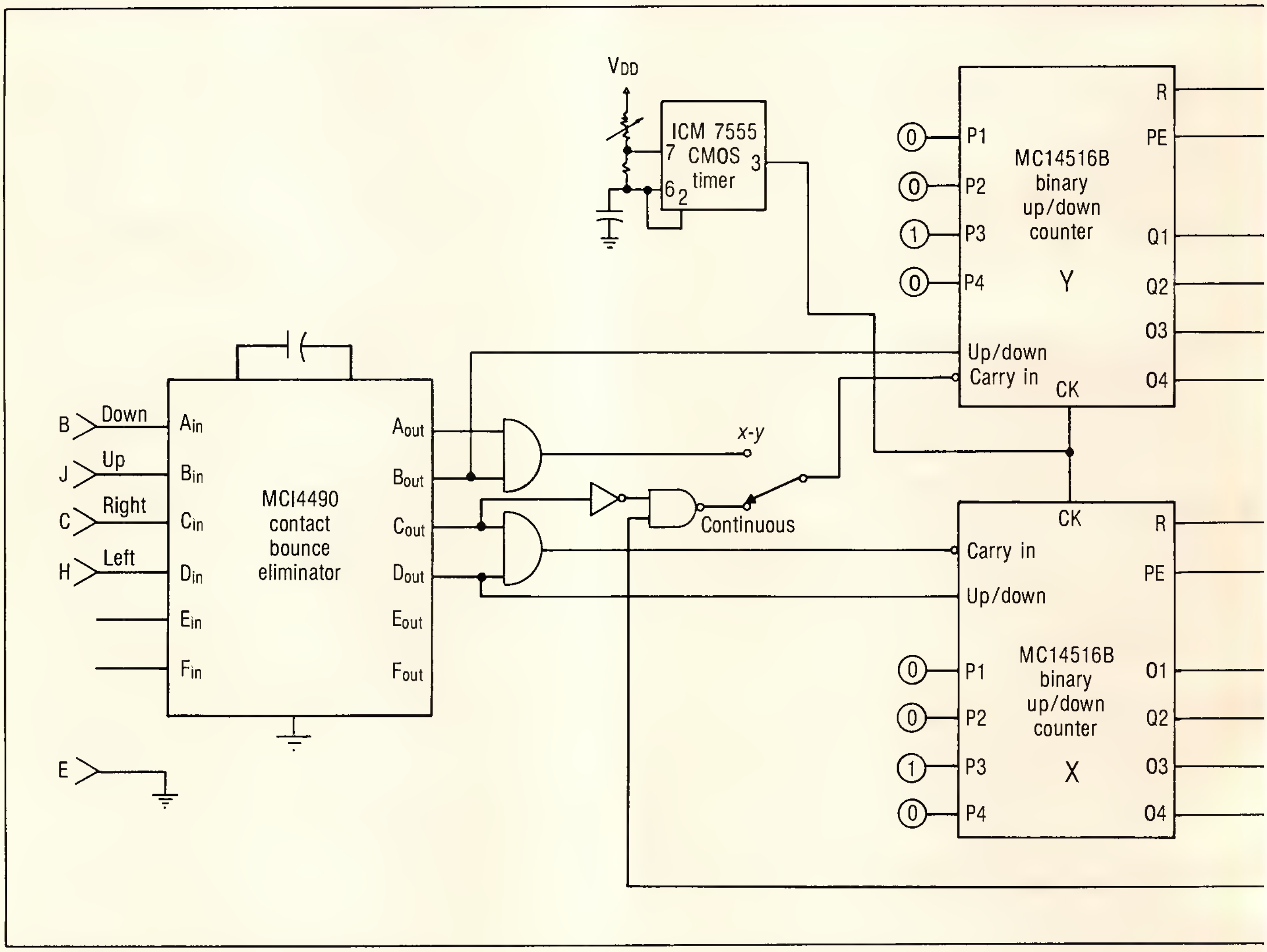


Figure 9. Simplified circuit diagram for the 5 × 5 array (routine connections omitted).

tion input to the filter is sampled and filtered at a rate of 200 speech samples per frame. This results in an 8-kHz sample rate which, according to Shannon's sampling theorem, provides a useful speech bandwidth of 4 kHz.

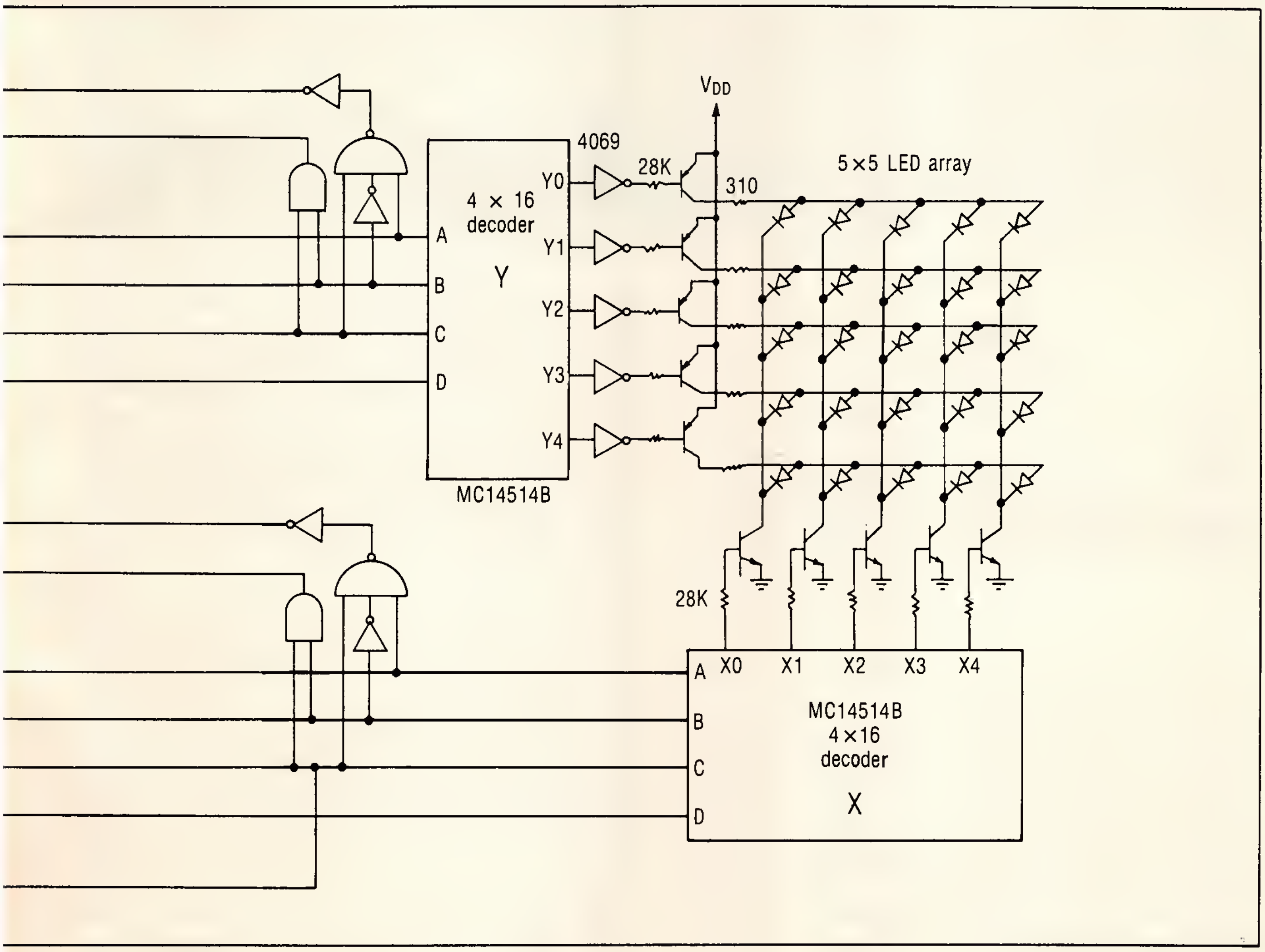
The lattice filter reflection coefficients, as outlined above, are generated at the speech analysis stage by the speech development system. The route to acquiring the set of reflection coefficients for each 25-ms frame involves, initially, the use of a linear prediction algorithm which tries to approximate any speech sample within a 25-ms analysis frame as a weighted linear combination of the P most immediate past samples of the signal (P = 10 here):

$$\hat{S}_n = -\sum_{k=1}^{P} a_k S_{n-k} \qquad (1)$$

where $\hat{S}_n$ is the predicted estimate of the actual current speech sample, S_n. We then form the expression

$$e_n = S_n - \hat{S}_n = \sum_{k=0}^{P} a_k S_{n-k}\,, \quad a_0 = 1, \qquad (2)$$

which indicates the error incurred between the current speech sample and its predicted value. To find the values for the prediction error coefficients ($a_1, a_2, \ldots, a_P$), the mean square error is minimized with respect to each signal sample in the analysis frame. The reflection coefficients (K1,K2,...,K10) are obtained finally by applying a transformation to the prediction error coefficients to yield an unconditionally stable lattice filter representation of the vocal tract. The reflection coefficients are therefore referred to as linear predictive coding, or LPC, coefficients. The output of the 14-bit digital filter is truncated to ten bits and passed on to an 8-bit D/A con-



verter every 125 ms (8-kHz sample rate). The output of the D/A converter at pin 8, SPEAKER, is a digitally controlled current source. A resistor must be used to convert the current to voltage. The ten-bit truncated filter output is available serially at pin 9, I/O.

The 5220 requires +5V, −5V, and ground supply voltages. Most of the chip operates between +5V and −5V. The digital inputs and outputs operate between +5V and ground.

The TMS 5220 VSP is very versatile, providing for NOP, READ BYTE, SPEAK EXTERNAL, READ AND BRANCH, LOAD ADDRESS, SPEAK, and RESET instructions. An internal status register includes talk status, buffer low, and buffer empty bits. These instructions and the status register are communicated to and from the VSP over the 8-bit VSP data bus (almost, but not quite, TTL-compatible) using control lines $\overline{WS}$, $\overline{RS}$, and $\overline{READY}$. An $\overline{INT}$ pin that is activated by certain conditions in the status register is also provided. These 12 pins are the microprocessor interface to the host microprocessor system that maintains control over the 5220.

Two means of inputting the LPC speech data derived from the development system to the 5220 are provided for. First, the data can be stored in microprocessor memory space and entered into the VSP along the 8-bit data bus using the control lines. This is done with the SPEAK EXTERNAL instruction. A 16-byte FIFO buffer is provided on the 5220 to accept data from the microprocessor. Second, the SPEAK and LOAD ADDRESS commands can be used, along with one or more TMS 6100 Voice Synthesis Memories (VSMs). The TMS 6100 is a PMOS 16K-byte ROM containing LPC speech data. Seven 5220 VSP pins provide for the VSP-VSM interface. These are ADD1, ADD2, ADD4, ADD8/DATA, ROMCLK, and M0 and M1. The latter three lines are clock and instruction control lines from the VSP. The others are address output lines, with the bidirectional line ADD8 also serving as a serial LPC data input.

The 6100 VSM differs from conventional ROMs in that it contains a 14-bit address counter and a 4-bit chip decode circuit (16 VSMs can be used). The 18-bit starting address of a word in the VSM is passed to the address counter in the VSM by executing five LOAD ADDRESS instructions (4 bits are passed along address bus ADD1, 2, 4, 8 at a time). When the SPEAK command is executed, the byte at the starting address is passed to the VSP through ADD8/DATA in bit-serial fashion. The address counter is automatically incremented, and bytes are passed to the VSP until a STOP code (ENERGY = 1111) is executed by the VSP. The use of a VSM with the 5220 VSP thus frees the microprocessor from handling LPC data.

Reference

1. G. Helms and S. Peterson, "Portable Speech Development System Creates Linear Predictive Codes," *Electronics,* Sept. 8, 1982, pp. 151-156.

Table 2.
Mapper EPROM.

Address	Mapper data
\$8000	Starting address low (WORD CODE)
\$8001	Starting address high (WORD CODE)
\$8002	Number of bytes low (WORD CODE)
\$8003	Number of bytes high (WORD CODE)

matic. A complementary transistor pair is used to provide the necessary level shifting.

The TMS 5220 can accept eight command codes. Only SPEAK EXTERNAL, command code \$60, and RESET, command code \$FF, are necessary for this application. The RESET command places the VSP in a known state. The SPEAK EXTERNAL command readies the VSP to accept speech data from the speech EPROMs. The speech data are directed to an internal 16-byte FIFO buffer from which the VSP takes data to perform speech calculations. The VSP also contains a status register that can be monitored to assess the VSP's status. The talk status bit (TS) is high when the VSP is talking. A buffer low flag (BL) is asserted (high) when eight bytes of FIFO are empty. Finally, the buffer empty flag (BE) indicates when the FIFO is empty. An $\overline{INT}$ pin on the VSP is useful for monitoring changes in these flags. Three additional VSP pins are used for control of the VSP data bus; these pins are $\overline{RS}$, $\overline{WS}$, and $\overline{READY}$. In this application, $\overline{READY}$ indicates when data are ready on the data bus.

Table 3 gives control statements describing VSP operation for the SPEAK EXTERNAL command. After a RESET command is executed, TS = 0, BL = 1, and BE = 1. When the SPEAK EXTERNAL command is issued, speech data are routed to the VSP FIFO by the 6502 microprocessor. The third equation in the table shows that BL is cleared after the first nine bytes have been written to the FIFO. The next equation shows that this one-to-zero transition in BL (denoted by $BL\downarrow$) sets the TS flag, starting speech calculations with the data in the FIFO. The positive edge of the BE flag clears TS, speech halts, and the VSP awaits another SPEAK EXTERNAL command.

The next two equations show the behavior of the $\overline{INT}$ output pin. This pin can be polled or used as an interrupt to indicate either that the FIFO buffer is low ($BL\uparrow$)

Table 3.
Flag action for SPEAK EXTERNAL.

Status register flags
BE = Boolean(FIFO empty)
Boolean(8 bytes FIFO empty): $BL \leftarrow 1$
Boolean(9 bytes FIFO full): $BL \leftarrow 0$
$BL\downarrow$: $TS \leftarrow 1$
$BE\uparrow$: $TS \leftarrow 0$
$\overline{INT}$ output pin
Boolean(read a status register byte) + Boolean(write a RESET byte): $\overline{INT} \leftarrow 1$
$TS\downarrow + BL\uparrow$: $\overline{INT} \leftarrow 0$
$\overline{WS}$ and $\overline{RS}$ input pins and $\overline{READY}$ output pin
$\overline{WS}\cdot\overline{\overline{RS}}$: $\overline{READY} \leftarrow 1$; VSP data bus = VSP status register outputs; if(data is stable) then $\overline{READY} \leftarrow 0$.
$\overline{\overline{WS}}\cdot\overline{RS}$: $\overline{READY} \leftarrow 1$; VSP data bus = 6520 PIA port B outputs; if(PIA data latched into either FIFO or command register along VSP data bus) then $\overline{READY} \leftarrow 0$.
$\overline{WS}\cdot\overline{RS}$: VSP data bus high impedance.

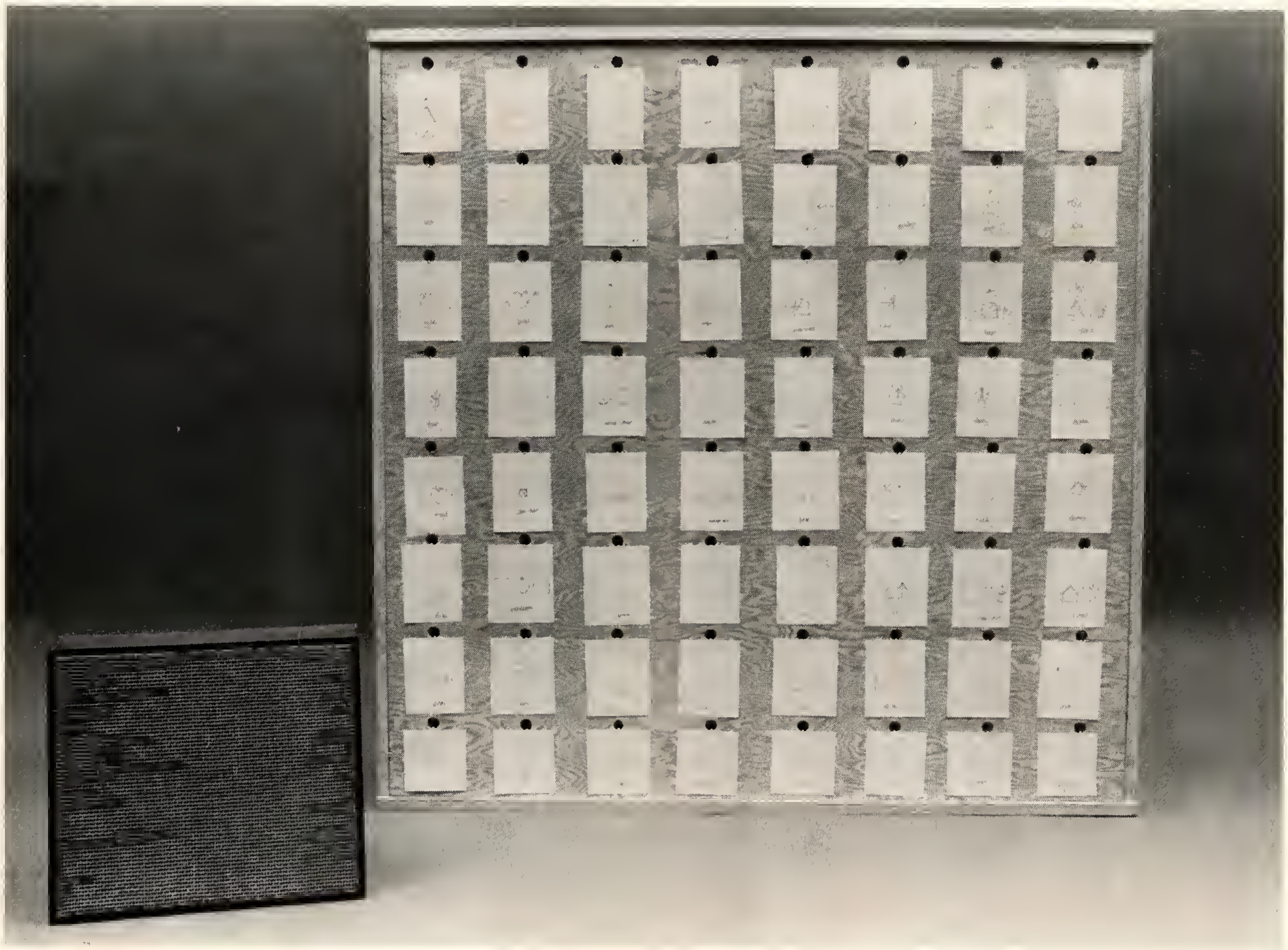


Figure 10. The 8 × 8 scanner.

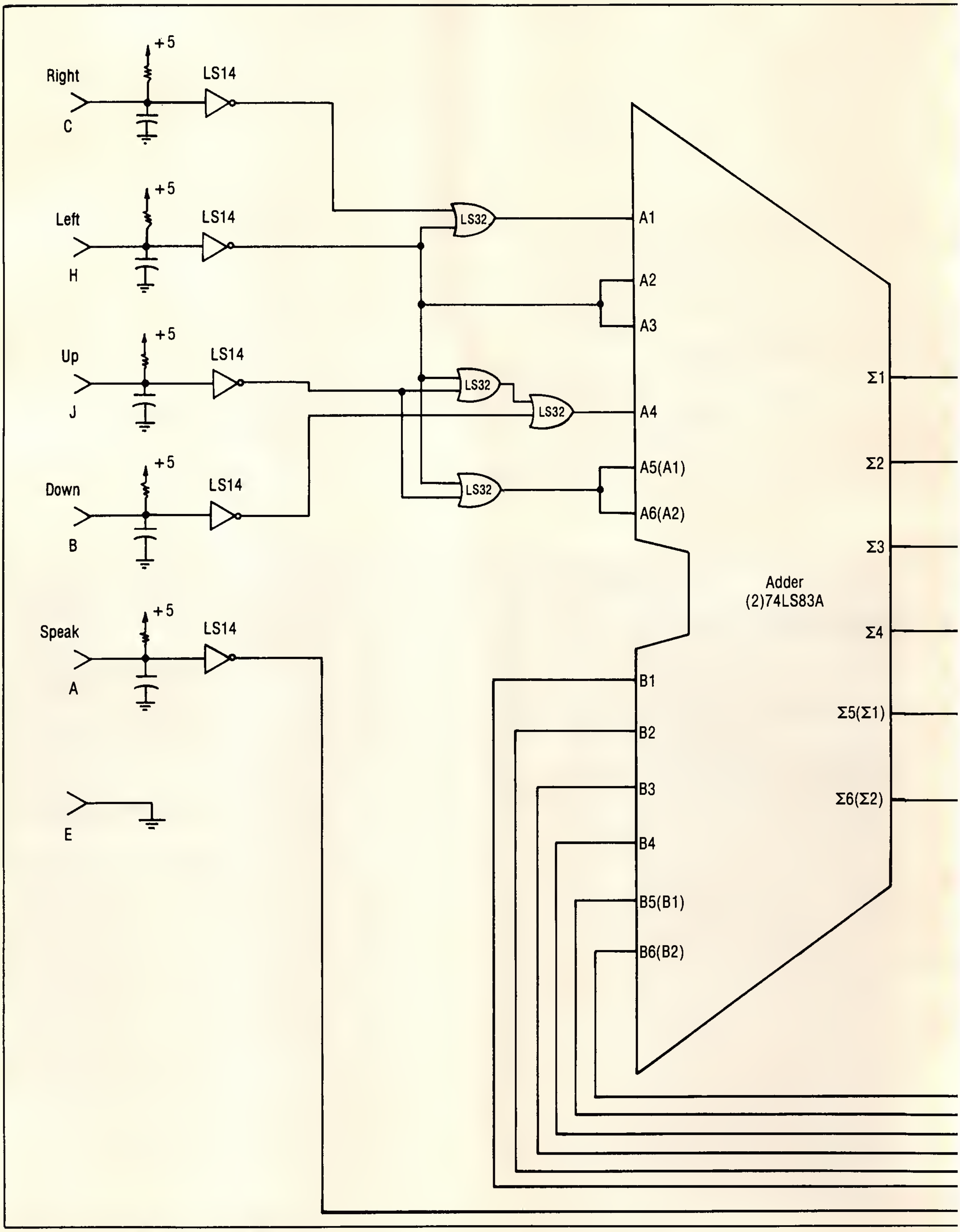
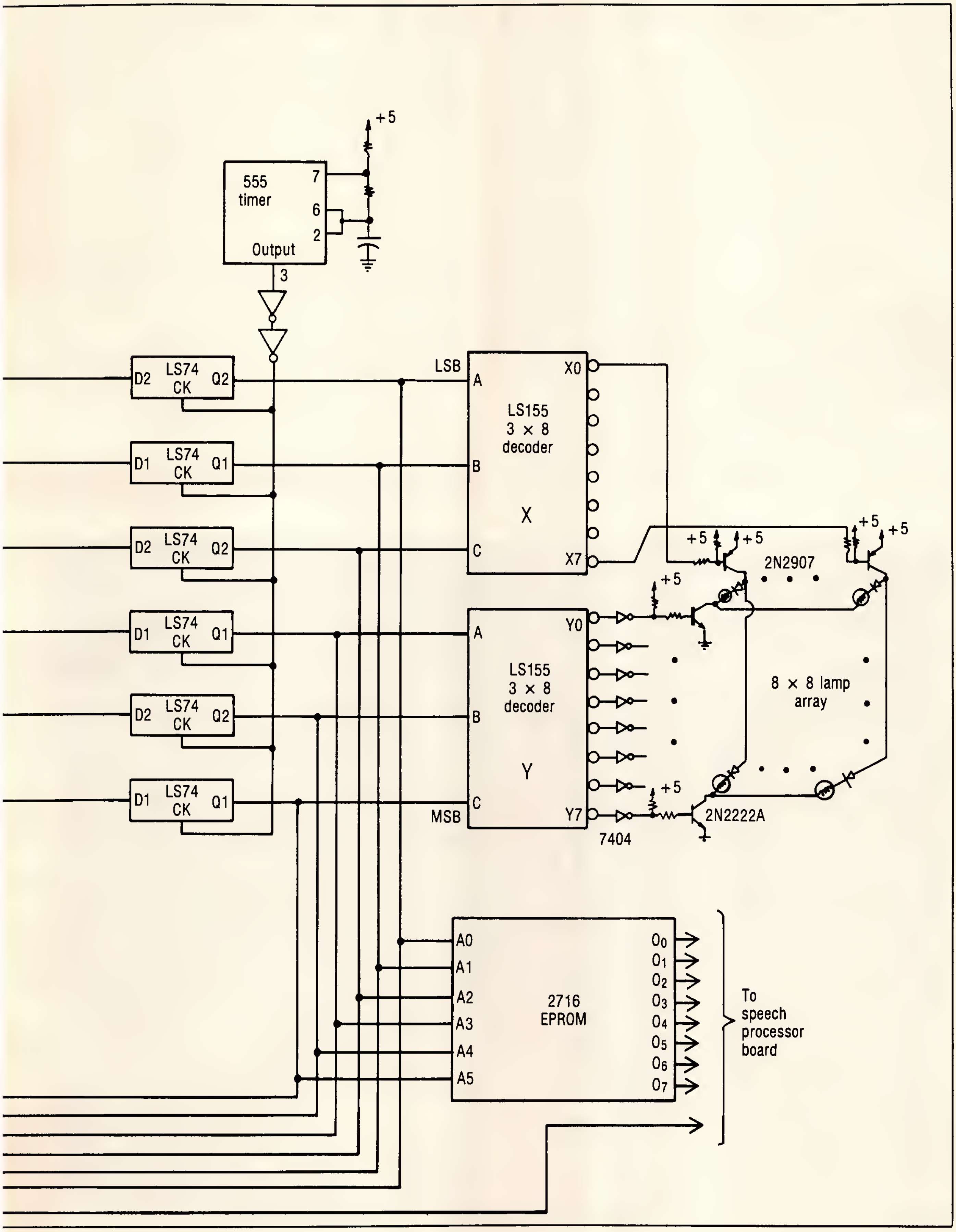


Figure 11. Simplified circuit diagram for the 8 × 8 array (routine connections omitted).


+5
555
timer
7
6
2
Output
3
D2
LS74
CK
Q2
D1
LS74
CK
Q1
D2
LS74
CK
Q2
D1
LS74
CK
Q1
D2
LS74
CK
Q2
D1
LS74
CK
Q1
LSB
A
B
C
LS155
3 × 8
decoder
X
X0
X7
+5
+5
+5
+5
2N2907
+5
A
B
C
MSB
LS155
3 × 8
decoder
Y
Y0
Y7
7404
8 × 8 lamp
array
+5
2N2222A
A0
A1
A2
A3
A4
A5
2716
EPROM
O0
O1
O2
O3
O4
O5
O6
O7
To
speech
processor
board

or that the VSP is finished speaking (TS↓). Since the microprocessor can fill the FIFO much faster than speech calculations can deplete it, only eight bytes should be written to the FIFO when $\overline{INT}$ goes low. Other than RESET, the only practical way to set $\overline{INT}$ to 1 after a buffer low interrupt is by reading the status register, as shown in the table. Note that since $\overline{INT}$ is set to zero on the positive transition of BL, $\overline{INT}$ can be set back to one either before or after new data are sent to the FIFO.

The next group of statements shows the operation of the $\overline{WS}$, $\overline{RS}$, and $\overline{READY}$ pins. The first statement shows that when $\overline{RS}$ (read select) is brought low, $\overline{READY}$ goes high until the VSP status register contents are stable on the data bus, at which time it returns low. The second statement shows that when $\overline{WS}$ (write select) is brought low, $\overline{READY}$ goes high until the 6520 data are latched into the command register, or into the FIFO if the last command was SPEAK EXTERNAL.

The 6520 PIA. From the schematic diagram in Figure 13, one can see that the 6520 is in addresses \$9000-\$9003. We follow the notation of an earlier paper[16]: the control registers are denoted by AC and BC, and the data direction registers by AD and BD. Port B is used for the 5220 data bus. Note that D7 of the VSP is the LSB. 6520 control lines CA1, CA2, and CB1 are programmed for negative edge triggering. The equations[16] that apply are

$$\begin{array}{rlrl} \mathrm{CA2}\downarrow: & \mathrm{AC6} \leftarrow 1 & \mathrm{CA1}\downarrow: & \mathrm{AC7} \leftarrow 1 \\ \mathrm{RA}: & \mathrm{AC6} \leftarrow 0 & \mathrm{RA}: & \mathrm{AC7} \leftarrow 0 \\ & & \mathrm{CB1}\downarrow & \mathrm{BC7} \leftarrow 1 \\ & & \mathrm{RB}: & \mathrm{BC7} \leftarrow 0 \end{array}$$

RA and RB are the read port A and read port B flip-flops.[16] Since CA1 = $\overline{READY}$, CA2 = SPEAK, and CB1 = $\overline{INT}$, one has

$$\begin{array}{rlrl} \mathrm{SPEAK}\downarrow: & \mathrm{AC6} \leftarrow 1 & \overline{\mathrm{READY}}\downarrow: & \mathrm{AC7} \leftarrow 1 \\ \mathrm{RA}: & \mathrm{AC6} \leftarrow 0 & \mathrm{RA}: & \mathrm{AC7} \leftarrow 0 \\ & & \overline{\mathrm{INT}}\downarrow: & \mathrm{BC7} \leftarrow 1 \\ & & \mathrm{RB}: & \mathrm{BC7} \leftarrow 0 \end{array}$$

Thus, the negative transitions on the $\overline{READY}$, SPEAK, and $\overline{INT}$ lines can be polled with AC7, AC6, and BC7,

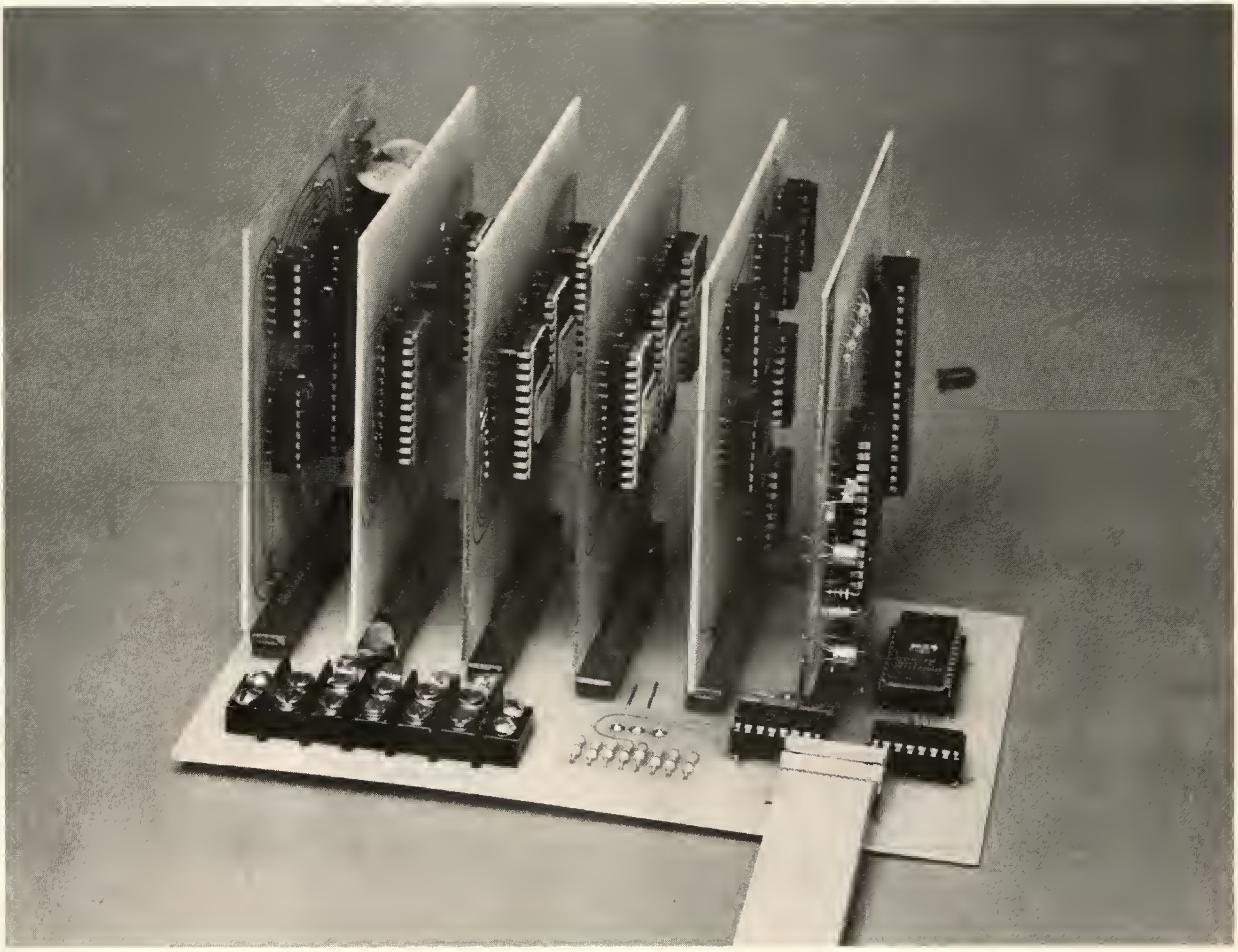


Figure 12. NMRC's speech synthesizer.

respectively. Furthermore, since Table 3 shows $TS\downarrow + BL\uparrow: \overline{INT} \leftarrow 0$, one has

$$TS\downarrow + BL\uparrow: BC7 \leftarrow 1$$

This means BC7 is set when the FIFO buffer is low or when the VSP is finished talking.

6502 program. On RESET, the 6502 executes instructions starting at location $F800. The program which controls the speech processor is given in Appendix B. Flowcharts for VSP initialization, for reading a byte, and for writing a byte are given elsewhere[14] and so are not given here. Three variables (six bytes) are located in zero page. These are SAL, SAH (starting address low and

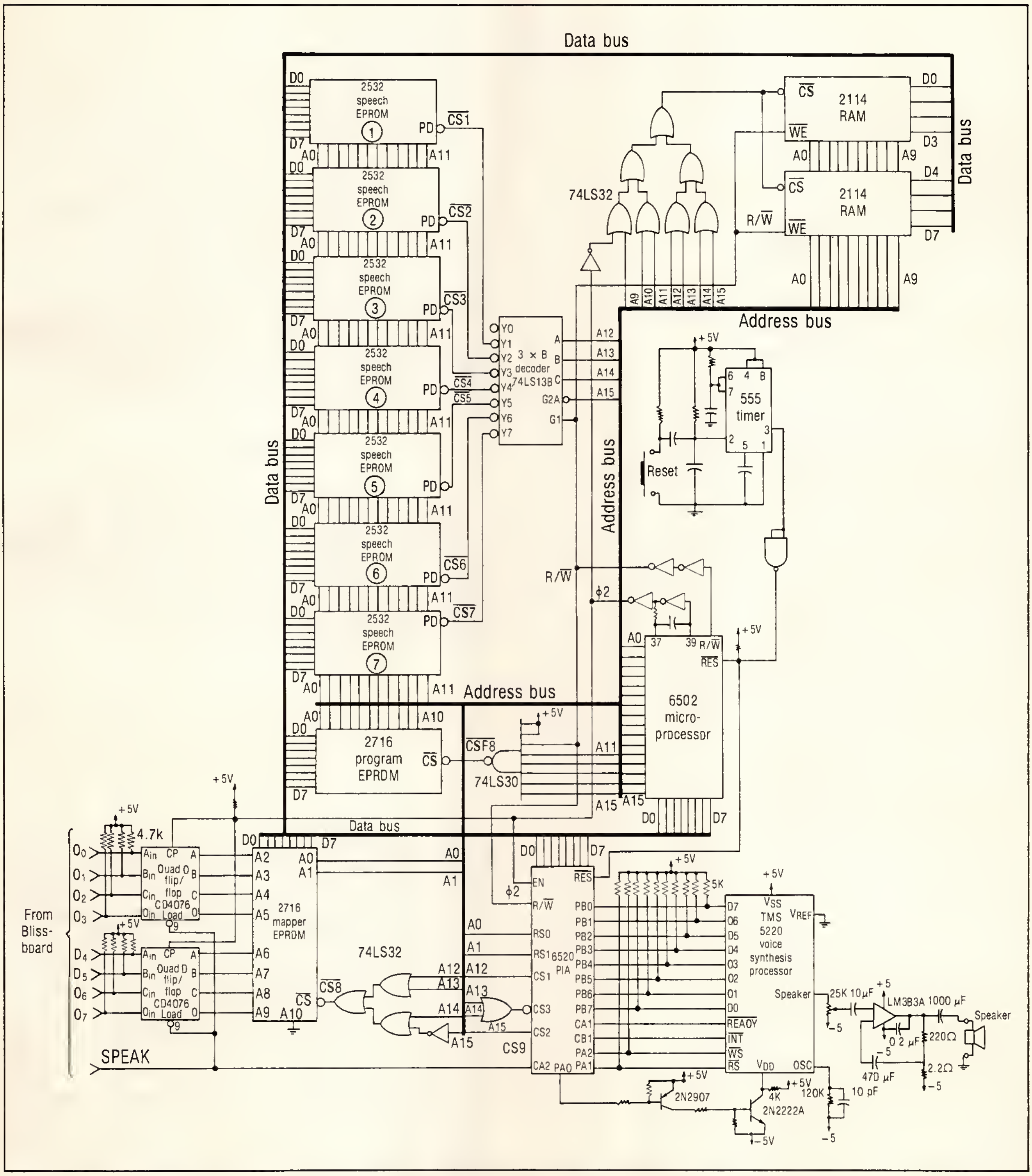


Figure 13. Simplified diagram for the speech circuit (routine connections omitted).

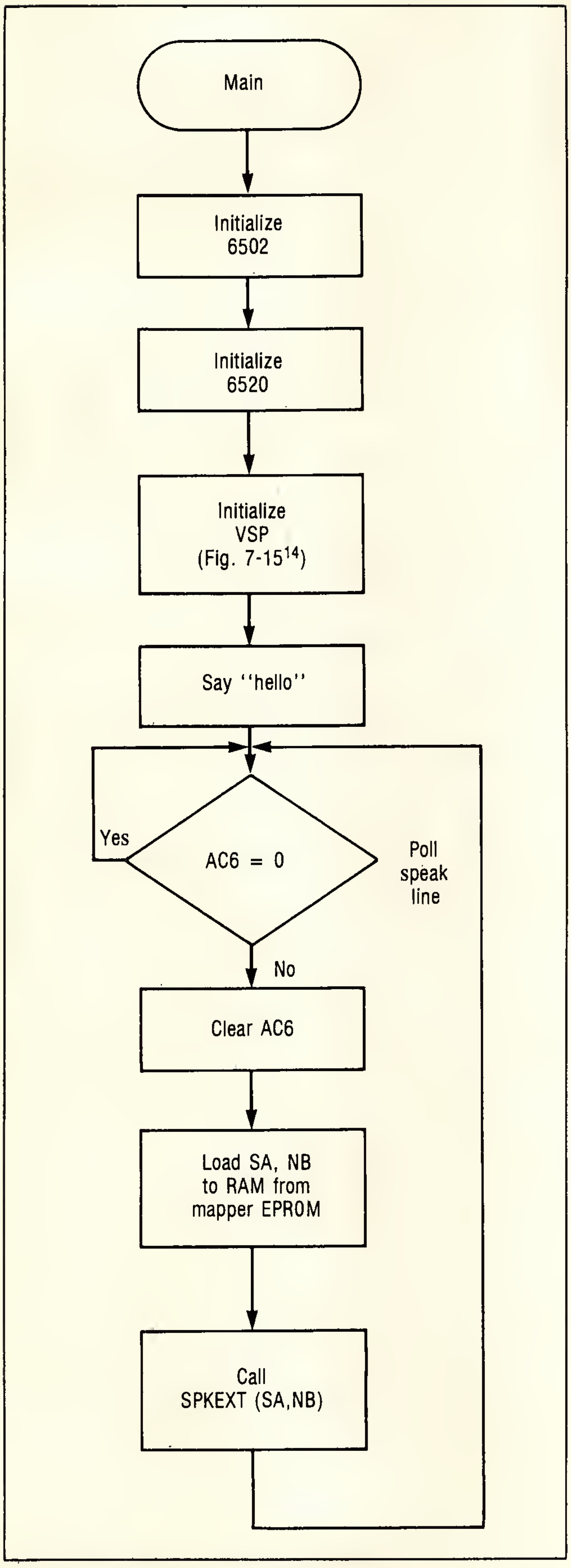


Figure 14. Main program at $F800.

high) = SA; NBL, NBH (number of bytes low and high) = NB; and IL, IH (index I low and high) = I. The low byte comes first because of the way the 6502 operates. A flowchart for the main program is shown in Figure 14. AC6 is polled until the SPEAK lines goes low, thereby setting AC6 to 1. AC6 is cleared, the starting address and number of bytes are read from the mapper EPROM, and subroutine SPKEXT (SPEAK EXTERNAL) is called. After the word is synthesized, the SPEAK line is again polled.

A flowchart for the subroutine SPKEXT is shown in Figure 15. Inputs to the subroutine are the starting address SA and the number of bytes in the word NB. If the starting address is $FFFF, then a word is not synthesized. PA4 is pulsed instead. This is a no-operation (NOP) code. Otherwise, the SPEAK EXTERNAL command is written to the VSP followed by the first 16 bytes of LPC speech data. This fills the FIFO. Then, BC7 ($\overline{INT}$) is polled until the FIFO buffer is low. When this occurs, eight more bytes are sent to the FIFO and BC7 ($\overline{INT}$) is again polled. This continues until all NB bytes of the word have been sent to the VSP. A byte of $00 is then sent to the FIFO and BC7 ($\overline{INT}$) is polled. The first interrupt on $\overline{INT}$ comes from the positive edge on BL. This sets BC7 since $TS\downarrow + BL\uparrow: \overline{INT} \leftarrow 0$, and

$$\overline{INT}\downarrow: BC7 \leftarrow 1$$

The program clears BC7 and resets $\overline{INT}$ to 1. Since TS = 1 (the VSP is still talking), the program polls BC7 ($\overline{INT}$) again. The second interrupt comes from the negative edge on the TS flag, meaning that the VSP is finished processing speech. Since TS = 0, the subroutine returns control to the calling program.

Our experience has indicated that the need for custom-designed equipment for the retarded and handicapped can be met by electrical engineering students working in a design class. The students provided this very specialized equipment for retarded and handicapped residents who did not have the funds or resources to have special units built specifically for them. Thus, benefits were received by residents and students alike—a viable means of communication for NMRC's nonverbal residents at minimal cost, and a valuable experience in the fields of electrical and rehabilitation engineering for the students.

Preliminary tests with several of NMRC's nonvocal residents have been successful. Further training and testing need to be performed. Figure 16 shows a resident

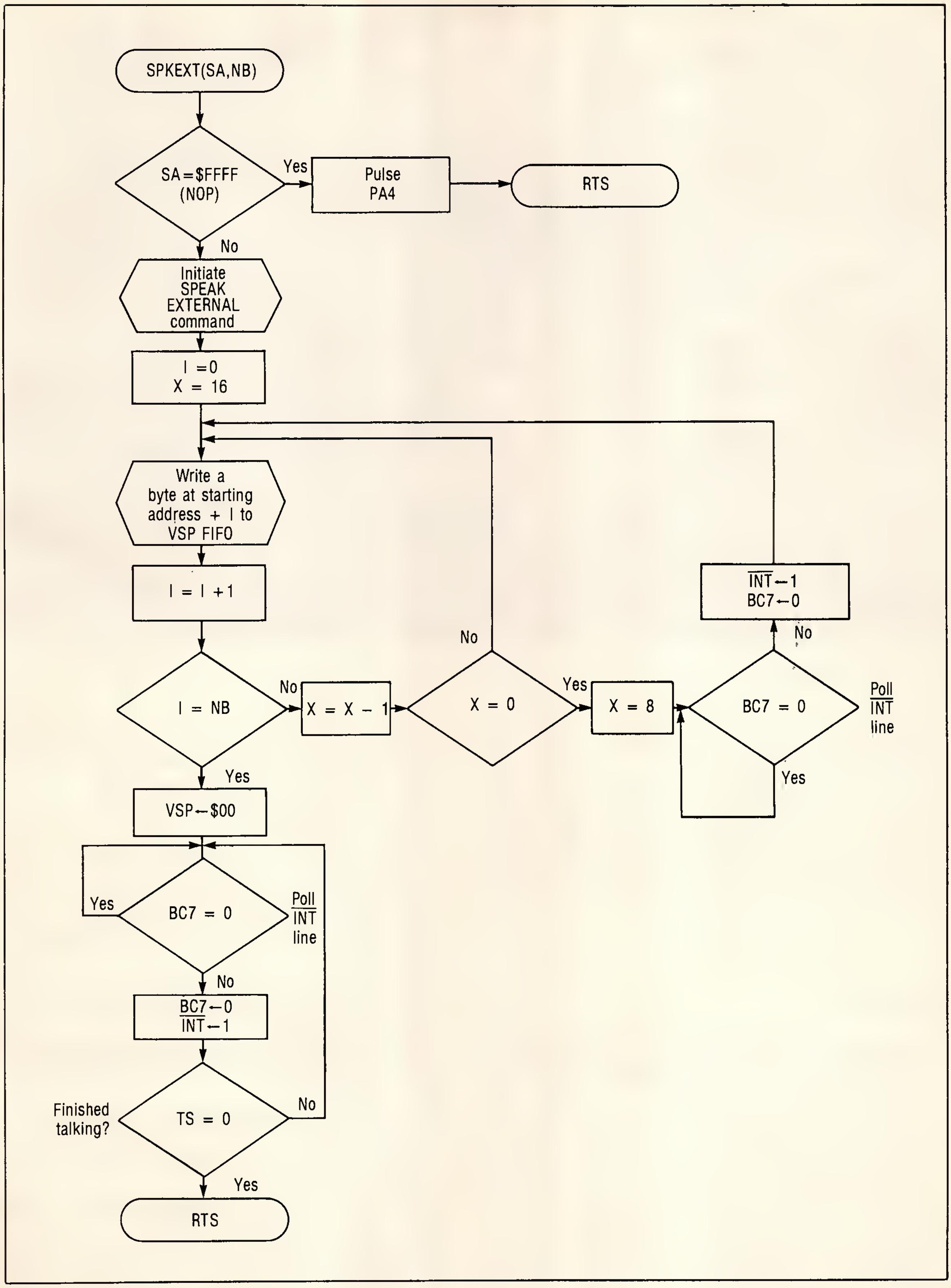


Figure 15. SPEAK EXTERNAL subroutine.

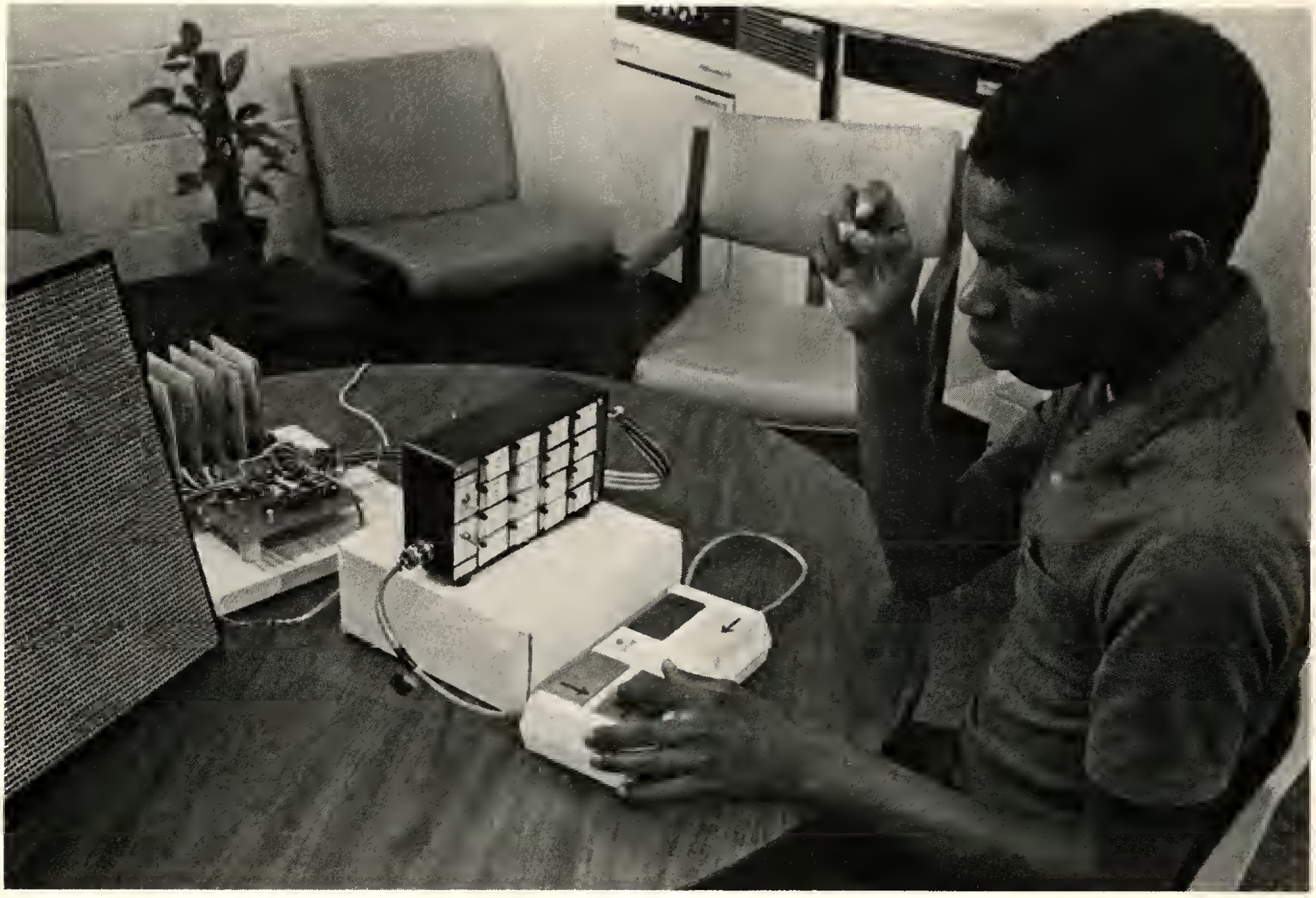


Figure 16. NMRC resident using 4 × 4 scanner. (Photo by John Tettleton)

using the 4 × 4 scanner. A safe package for the speech processor/power supply will soon be developed.

For those with plenty of funds, software that displays 500 Blissymbols is available[17] for the Apple II microcomputer. This system can replace the inexpensive Blissboards described here, but at a much greater cost than our scanner. Our custom control and speech synthesizer can still be used with such a system. ■

References

1. "The Problem of Mental Retardation," US Department of Health, Education, and Welfare, President's Committee on Mental Retardation, Washington, DC.
2. N. M. Robinson and H. B. Robinson, *The Mentally Retarded Child: A Psychological Approach,* McGraw-Hill, New York, 1976.
3. G. Vanderheiden and K. Grilley, *Non-vocal Communication Techniques and Aids for the Severely Physically Handicapped,* University Park Press, Baltimore, 1977.
4. E. R. Ritvo, "Definition of Syndrome of Autism," The National Society for Autistic Children.
5. C. Van Riper, *Speech Correction: Principles and Methods,* Prentice-Hall, Englewood Cliffs, NJ, 1972.
6. E. T. McDonald, *Teaching and Using Blissymbolics,* The Blissymbolics Communication Institute, Toronto, 1980.
7. McDonald, *Teaching and Using Blissymbolics.*
8. C. K. Bliss, "Mr. Symbol Man" (film), Sydney, Australia, 1975.
9. McDonald, *Teaching and Using Blissymbolics.*
10. Vanderheiden and Grilley, *Non-vocal Communication Techniques.*
11. Vanderheiden and Grilley, *Non-vocal Communication Techniques.*
12. R. L. Schiefelbush, ed., *Nonspeech Language and Communication—Analysis and Intervention,* University Park Press, Baltimore, 1980.
13. J. B. Eulenberg and J. Rosenfeld, "Vocaid—A New Product from Texas Instruments," *RESNA Newsletter*, Vol. 1, No. 2, Aug. 1982.
14. *Speech Synthesis Technology Student Guide,* Texas Instruments, 1980.
15. *TMS 5220 Voice Synthesis Processor Data Manual,* Texas Instruments, 1981.
16. D. F. Hanson, "An Improved Model for a Microcomputer Component—The 6520 PIA," *IEEE Micro,* Vol. 1, No. 4, Nov. 1981, pp. 17-25.
17. "Blissymbolics: Blissboard," Minnesota Educational Computing Consortium, 2520 Broadway Drive, St. Paul, MN 55113.

Appendix A—Word codes for the NMRC speech synthesizer

Word code	Word	Address	Bytes
Nouns and pronouns			
00	I	1000	0074
01	you	1074	0074
02	mama	10E8	00AD
03	daddy	1195	0078
04	brother	120D	0081
05	sister	128E	0077
06	friend	1305	0078
07	teacher	137D	006C
08	boy	13E9	0065
09	girl	144E	0073
0A	therapist	14C1	0099
0B	grandmother	155A	00B8
0C	baby	1612	008A
0D	doctor	169C	0085
Places			
0E	classroom	1721	00AE
0F	cottage	17CF	0073
10	home	1842	006E
11	canteen	18B0	00B3
12	pool	1963	0066
13	gym	1AA6	005F
14	music	1B05	008E
15	library	1B93	00BF
16	park	1C52	0074
17	auditorium	1CC6	0111
Verbs			
18	eat	1DD7	0051
19	drink	1E28	0065
1A	go	1E8D	0068
1B	want	1EF5	006D
1C	have	2000	0066
1D	give	2066	0071

1E	write	20D7	0060
1F	make	2137	005E
20	need	2195	0087
21	see	221C	0081
22	work	229D	0068
23	am	2305	006C
24	is	2371	0075
25	are	23E6	0058
26	get	243E	0052
27	know	2490	0074
28	sleep	2504	007B
29	do	257F	005E
2A	keep	25DD	003E
2B	sing	261B	0078
2C	sit	2693	0063
2D	stand	26F6	0094
2E	stop	278A	0079
2F	think	2803	006F
30	understand	2872	00E0
31	wish	2952	0075

Objects

32	desk	29C7	0062
33	pencil	2A29	0067
34	paper	2A90	0082
35	book	2B12	0056
36	toilet	2B68	008F
37	token	2BF7	008F
38	soap	2C86	0072
39	toothbrush	2CF8	009D
3A	toothpaste	2D95	00AD
3B	clothing	2E42	00AE
3C	bus	3000	0059
3D	record	3059	008C
3E	comb	30E5	0073
3F	table	3158	0063
40	TV	31BB	0095
41	bath	3250	005C
42	wheelchair	32AC	00BF
43	telephone	336B	00B0
44	chair	341B	006E
45	glass	3489	0083
46	party	350C	0078
47	radio	3584	00D6
48	letter	365A	0077
49	symbol	36D1	00A0

Positive-negative indicators

4A	yes	3771	006B
4B	no	37DC	005E

Social words

4C	hello	3910	007B
4D	goodbye	398B	0092
4E	please	3AB2	00A8

Adjectives

4F	happy	3C28	0078
50	sad	3CA0	006B
51	angry	3D0B	009A
52	good	3DA5	007B
53	bad	3E20	0076
54	yuk!	3E96	005C
55	sick	3EF2	0066
56	tired	3F58	0073
57	hungry	4000	008B
58	cold	408B	0095
59	smart	4120	0071
5A	stupid	4191	0090
5B	big	4221	006F
5C	little	4290	0078
5D	pretty	4308	0061
5E	funny	4369	0068
5F	nice	43D1	0092
60	ugly	4463	008E

Occasions

61	birthday	44F1	009B
62	Christmas	458C	009E
63	Easter	462A	0075
64	Thanksgiving	469F	00CB
65	Valentine's Day	476A	0145

Additional verbs

66	swim	48AF	0093
67	run	4942	0081

68	play	49C3	0077
69	dance	4A3A	007A
6A	love	4AB4	007A
6B	count	4B2E	006A
6C	jump	4B98	007A
6D	help	4C12	006B
6E	listen	4C7D	006D
6F	talk	4CEA	0067
70	find	4D51	00B6
71	fix	4E07	009E
72	forget	4EA5	0062
73	brush	4F07	0069
74	cook	4F70	0057
75	cry	5000	0067
76	point	5067	006E
77	rain	50D5	0090
78	send	5165	0089
79	shave	51EE	00A2
7A	paint	5290	005B
7B	pay	52EB	0067
7C	stay	5352	007A
7D	turn	53CC	0081
7E	visit	544D	0088
7F	win	54D5	0069

Negatives

80	not	553E	0083
81	never	55C1	008F

Clothing

82	shirt	5650	0075
83	shoes	56C5	009A
84	pants	575F	0063
85	coat	57C2	005C
86	socks	581E	0092

Time

87	today	58B0	008A
88	tomorrow	593A	00A8
89	yesterday	59E2	00C7
8A	morning	5AA9	00C3
8B	night	5B6C	0073

Interrogatives

8C	who?	5BDF	0098
8D	what?	5C77	0079
8E	where?	5CF0	006C
8F	why?	5D5C	0082

Days of the week

90	Sunday	5DDE	00A1
91	Monday	5E7F	00AA
92	Tuesday	5F29	00AB
93	Wednesday	6000	0097
94	Thursday	6097	00B4
95	Friday	614B	0085
96	Saturday	61D0	00AB

Colors

97	red	627B	007A
98	blue	62F5	0079
99	black	636E	0085
9A	white	63F3	006B
9B	green	645E	00AE
9C	yellow	650C	0094

Prepositions

9D	in	65A0	0074
9E	on	6614	005B
9F	under	666F	0072
A0	to	66E1	0050
A1	at	6731	0052

Pronouns

A2	he	6783	0063
A3	she	67E6	006F
A4	they	6855	0090
A5	we	68E5	0067
A6	it	694C	0059
A7	them	69A5	00A3

Months of the year

A8	January	6A48	00D6
A9	February	6B1E	00BC
AA	March	6BDA	0088
AB	April	6C62	007B
AC	May	6CDD	00A1
AD	June	6D7E	0072
AE	July	6DF0	009F
AF	August	6E8F	008D
B0	September	6F1C	00A6
B1	October	7000	0096
B2	November	7096	00C6
B3	December	715C	009D

Phrases

B4	token store	19C9	00DD
B5	record player	2EF0	00E1
B6	don't know	383A	00D6
B7	thank you	3A1D	0095
B8	you're welcome	3B5A	00CE
B9	I'm fine	71F9	00B5
BA	I need some help	737F	0128
BB	Where are you going?	74A7	010B
BC	What do you want?	75B2	00E9
BD	How are you?	72AE	00D1

No operation (NOP)

BE	FFFF	FFFF
BF	FFFF	FFFF
C0	FFFF	FFFF
.		
.		
.		
FE	FFFF	FFFF
FF	FFFF	FFFF

190 words and phrases total.

Appendix B—6502 program listing

```
                                                  ;Appendix II.

                                              ;6502 Program Listing.

      F800                    *=$F800         ;Address of Program EPROM
      9000                    AD =$9000       ;6520 A Data Direction Register Add.
      9002                    BD =$9002       ;6520 B Data Direction Register Add.
      9001                    AC =$9001       ;6520 A Control Register Address
      9003                    BC =$9003       ;6520 B Control Register Address
      9000                    PA =$9000       ;6520 Port A Address
      9002                    PB =$9002       ;6520 Port B Address
      8000                    MAPPER =$8000   ;Mapper EPROM Address
      0000                    SAL =$0000      ;Starting Address Low (Appendix I)
      0001                    SAH =$0001      ;Starting Address High (Appendix I)
      0002                    NBL =$0002      ;Number of Bytes Low (Appendix I)
      0003                    NBH =$0003      ;Number of Bytes High (Appendix I)
      0004                    IL =$0004       ;Index I Low
      0005                    IH =$0005       ;Index I High
                     ;---------------------------------------------------------------------
F800  D8                      CLD            ;Initialize 6502 Microprocessor
F801  18                      CLC
F802  78                      SEI
F803  B8                      CLV
F804  A2 FF                   LDX  #$FF
F806  9A                      TXS
                     ;---------------------------------------------------------------------
F807  A9 00                   LDA  #$00      ;Initialize 6520 PIA
F809  8D 01 90                STA  AC
F80C  8D 03 90                STA  BC
F80F  8D 02 90                STA  BD
F812  A9 FF                   LDA  #$FF
F814  8D 00 90                STA  AD
F817  A9 04                   LDA  #$04
F819  8D 01 90                STA  AC
F81C  8D 03 90                STA  BC
F81F  A9 06                   LDA  #$06      ;Power-up VSP (set PA0 = 0)
F821  8D 00 90                STA  PA
                     ;---------------------------------------------------------------------
F824  A2 09                   LDX  #$09      ;Initialize VSP using Figure 7-15[14].
F826  A9 FF          INITVS:  LDA  #$FF      ;Write nine RESETs to VSP
F828  20 88 F8                JSR  WRITEB
F82B  CA                      DEX
F82C  D0 F8                   BNE  INITVS
F82E  20 79 F8                JSR  DLA50     ;Wait 50ms.
F831  A9 FF                   LDA  #$FF      ;Write two RESETs to VSP
F833  20 88 F8                JSR  WRITEB
F836  A9 FF                   LDA  #$FF
F838  20 88 F8                JSR  WRITEB
F83B  20 B8 F8                JSR  READB     ;Clear INTbar line (pin 17)
F83E  29 E0                   AND  #$E0      ;  and Read Status Register Flags
F840  C9 60                   CMP  #$60
F842  F0 08                   BEQ  HELLO     ;Initialization Successful
F844  A9 0E          STUCK:   LDA  #$0E      ;Initialization Failed
F846  8D 00 90                STA  PA        ;PA3 = 1 failure indicator
F849  4C 44 F8                JMP  STUCK
F84C  A9 10          HELLO:   LDA  #$10      ;Set up Subroutine call for "Hello"
F84E  85 00                   STA  SAL       ; Word Code = $4C
F850  A9 39                   LDA  #$39      ; Starting Address = $3910
F852  85 01                   STA  SAH
F854  A9 7B                   LDA  #$7B
```

```
F856  85 02          STA  NBL        ;Number of Bytes = $007B
F858  A9 00          LDA  #$00
F85A  85 03          STA  NBH
F85C  20 E1 F8       JSR  SPKEXT     ;Say "Hello"
F85F  2C 01 90  SPEAK:   BIT  AC         ;Poll SPEAK line AC6 (6520)
F862  50 FB          BVC  SPEAK      ;No change
F864  AD 00 90       LDA  PA         ;Clear AC6
F867  A2 00          LDX  #$00       ;Say the word on the Blissboard
F869  BD 00 80  ADDRES:  LDA  MAPPER,X   ;Load SA, NB to RAM via A
F86C  95 00          STA  SAL,X      ; from MAPPER EPROM
F86E  E8             INX
F86F  E0 04          CPX  #$04
F871  D0 F6          BNE  ADDRES
F873  20 E1 F8       JSR  SPKEXT     ;Say the word
F876  4C 5F F8       JMP  SPEAK

                ;------------------------------------------------------------------
F879  A2 20     DLA50:   LDX  #$20       ;50 ms delay subroutine
F87B  A0 FF     XLOOP:   LDY  #$FF
F87D  A5 00     YLOOP:   LDA  SAL
F87F  A5 00          LDA  SAL
F881  88             DEY
F882  D0 F9          BNE  YLOOP
F884  CA             DEX
F885  D0 F4          BNE  XLOOP
F887  60             RTS

                ;------------------------------------------------------------------
                                 ;Write a byte Subroutine from Figure 7-13[14].
F888  48        WRITEB:  PHA             ;Data for VSP in A on calling WRITEB
F889  A9 00          LDA  #$00       ;Set 6520 Port B for outputs
F88B  8D 03 90       STA  BC         ;Select Data Direction Register B
F88E  A9 FF          LDA  #$FF       ;Configure Port B for Output
F890  8D 02 90       STA  BD
F893  A9 04          LDA  #$04       ;Select Port B
F895  8D 03 90       STA  BC
F898  68             PLA             ;Return Data for VSP to A
F899  8D 02 90       STA  PB         ;Force Data onto VSP Data Bus
F89C  A9 02          LDA  #$02       ;Set VSP WSbar = 0 (pin 27)
F89E  8D 00 90       STA  PA
F8A1  2C 01 90  READYW:  BIT  AC         ;Wait until VSP READYbar line = 0
F8A4  10 FB          BPL  READYW     ; (when AC7 of 6520 = 1)
F8A6  A9 06          LDA  #$06       ;Set WSbar (pin 27) = 1
F8A8  8D 00 90       STA  PA
F8AB  AD 00 90       LDA  PA         ;Clear AC7 and wait 12 us
F8AE  AD 00 90       LDA  PA
F8B1  AD 00 90       LDA  PA
F8B4  AD 00 90       LDA  PA         ;Finished
F8B7  60             RTS

                ;------------------------------------------------------------------
                                 ;Read a Byte Subroutine from Figure 7-14[14].
F8B8  A9 00     READB:   LDA  #$00       ;Set 6520 Port B for input
F8BA  8D 03 90       STA  BC
F8BD  8D 02 90       STA  BD
F8C0  A9 04          LDA  #$04       ;Port B input
F8C2  8D 03 90       STA  BC
F8C5  8D 00 90       STA  PA         ;Set VSP RSbar (pin 28) = 0
F8C8  2C 01 90  READYR:  BIT  AC         ;Wait until VSP READYbar = 0
F8CB  10 FB          BPL  READYR     ; (when AC7=1 on 6520)
F8CD  AD 02 90       LDA  PB         ;Input data from Port B
F8D0  48             PHA             ;Save data in A
F8D1  A9 06          LDA  #$06       ;Set RSbar (pin 28) = 1
```

```
F8D3   8D  00  90            STA  PA
F8D6   AD  00  90            LDA  PA          ;Clear AC7 and wait 10.5 us
F8D9   AD  00  90            LDA  PA
F8DC   AD  00  90            LDA  PA
F8DF   68                    PLA              ;Return with data in A
F8E0   60                    RTS
                    ;-------------------------------------------------------------
F8E1   D8           SPKEXT:  CLD
F8E2   A6  00                LDX  SAL         ;Is SA=$FFFF?
F8E4   E8                    INX
F8E5   D0  13                BNE  ISSUE
F8E7   A6  01                LDX  SAH
F8E9   E8                    INX
F8EA   D0  0E                BNE  ISSUE
F8EC   A9  16                LDA  #$16        ;Yes, SA=$FFFF
F8EE   8D  00  90            STA  PA          ; so pulse PA4 once
F8F1   20  79  F8            JSR  DLA50
F8F4   A9  06                LDA  #$06
F8F6   8D  00  90            STA  PA
F8F9   60                    RTS
F8FA   A9  60       ISSUE:   LDA  #$60        ;No, so Send VSP "SPEAK EXTERNAL"
F8FC   20  88  F8            JSR  WRITEB      ; Instruction, Op-Code $60
F8FF   A9  00                LDA  #$00        ;Index I = 0
F901   85  04                STA  IL
F903   85  05                STA  IH
F905   A2  10                LDX  #$10        ;Index X = 16
F907   A4  04       SHIPIT:  LDY  IL
F909   B1  00                LDA  (SAL),Y     ;Load byte at (SA)+I
F90B   20  88  F8            JSR  WRITEB      ;Write byte to FIFO
F90E   E6  04                INC  IL          ;I = I + 1
F910   D0  04                BNE  TESTI
F912   E6  05                INC  IH          ;IH = IH + 1
F914   E6  01                INC  SAH         ;SAH = SAH + 1
F916   A5  02       TESTI:   LDA  NBL         ;I = NB?
F918   C5  04                CMP  IL
F91A   D0  1B                BNE  TESTX       ;No, Test X
F91C   A5  03                LDA  NBH         ;IH = NBH?
F91E   C5  05                CMP  IH
F920   D0  15                BNE  TESTX       ;No, Test X
F922   A9  00                LDA  #$00        ;Yes, through sending word data
F924   20  88  F8            JSR  WRITEB      ;Send $00 to VSP
F927   2C  03  90   POLINT:  BIT  BC          ;Poll VSP INTbar (6520 BC7) line for
F92A   10  FB                BPL  POLINT      ; BL or Talk Status = through talking
F92C   AD  02  90            LDA  PB          ;Clear BC7
F92F   20  B8  F8            JSR  READB       ;Set INTbar
F932   29  80                AND  #$80        ;TS = 0?
F934   D0  F1                BNE  POLINT      ;No, talking
F936   60                    RTS              ;Yes, through talking so return
F937   CA           TESTX:   DEX              ;X = X-1
F938   D0  CD                BNE  SHIPIT      ;X not zero, so write another byte
F93A   A2  08                LDX  #$08        ;X=0, so set X = 8
F93C   2C  03  90   WAITBL:  BIT  BC          ;Poll VSP INTbar (6520 BC7) line
F93F   10  FB                BPL  WAITBL      ; until FIFO Buffer is Low
F941   AD  02  90            LDA  PB          ;Clear BC7
F944   20  B8  F8            JSR  READB       ;Set INTbar
F947   4C  07  F9            JMP  SHIPIT      ;Send more data to VSP
     FFFA                    *=$FFFA           ;RESET and Interrupt vectors
FFFA   00  F8  00            .BYTE $00,$F8,$00,$F8,$00,$F8
FFFD   F8  00  F8
     0000                    .END
```

University of Mississippi electrical and mechanical engineering students—contributing to the 1983 projects were, front row left to right, George McNeer, Charles Smith, Jr., Stephen Coleman, Thomas Ronaldi, and Eric Brenkert; back row left to right, Boon Tan, Gerald Lantz, Robert Logan, Jr., Steve Baker, and John Anderson. Not shown, Howard White. (Photo by Betty Brenkert and courtesy of the *Oxford Eagle*.)

Donald F. Hanson is an associate professor of electrical engineering at the University of Mississippi. His research interests include digital and analog electronics applications, including microprocessors and microcomputer interfacing, and the development of mathematical and numerical techniques for solving boundary-value problems of electromagnetic theory. He was previously an assistant professor of electrical engineering at Iowa State University, Ames.

Hanson received the BS, MS, and PhD degrees in electrical engineering from the University of Illinois, Urbana, in 1969, 1972, and 1976, respectively. He is a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu.

Questions about this article can be directed to Hanson at the University of Mississippi, Electrical Engineering Department, University, MS 38677.

Peggy Cook Power is presently employed by the North Mississippi Retardation Center in Oxford, Mississippi, where she serves as a speech/language pathologist and as the supervisor of communication services. She also conducts workshops and inservice training programs on Blissymbolics and other forms of augmentative communication. Power received a bachelor of communicative disorders degree from the University of Mississippi in 1979, and a master of speech pathology degree from the same institution in 1980.

This multiboard, Z80-based, X.25 DTE device achieves high data communications performance through a well-integrated hardware/software architecture.

A Multimicrocomputer-based Structure for Computer Networking

Alberto Faro, Orazio Mirabella, and Lorenzo Vita
University of Catania

For various technical, economic, and political reasons, distributed systems are coming into ever wider use. These systems consist of several computers interconnected through a communication subnetwork, or CS, in order to obtain a cooperation among processes resident in them. We can identify two major technical concerns in the design of distributed systems: the first involves the connection of the computers to the communication subnetwork with suitable associated software to obtain an efficient transport of information among the systems; the second involves the software package used to obtain cooperation among the processes resident in the systems.

An interesting architecture for distributed systems is the OSI—Open Systems Interconnection—model proposed by the ISO.[1] This architecture consists of seven layers:

- *Physical layer.* Defines the electrical, mechanical, functional, and procedural characteristics of the connections between interconnected computer systems.
- *Link layer.* Provides the means to transmit data between adjacent interconnected systems despite a relatively high error rate.
- *Network layer.* Provides the means to route information packets from a source to a destination system.
- *Transport layer.* Relieves the higher layers from any concern with how reliable and cost effective transfer of data is achieved.
- *Session layer.* Binds (and unbinds) interacting pro-

 0272-1732/85/0400-0053$01.00 © 1985 IEEE

cesses into a logical relationship that controls data exchange with respect to synchronization and structure.

- *Presentation layer.* Deals with the representation and manipulation of structured data for the benefit of the application layer.
- *Application layer.* Acts as a window between communicating user processes.

Each layer, or *n*-layer, offers specific services to the layer above it, and internally consists of communicating entities which exchange data units according to a set of rules called an *n*-protocol. Each *n*-entity sends these *n*-protocol data units to peer entities by using the services offered by a (*n*-1) distributed service provider, or DSP, at a service access point. The DSP has a multilayered internal structure in which each layer has the aforementioned structure (Figure 1). When an entity uses the services offered by two or more service providers, it is called a relay entity. Any communication system performing a relay function is called a relay system.

Relay systems are particularly useful for providing gateways between different CSs (Figure 2). Moreover, these systems can be used not only for communication purposes but also to test and control internal CS activity and the interfaces between different CSs.

We know that there are two chief ways to implement such relay systems.[2] The first one involves using a completely dedicated system outside the communication subsystems to be interconnected; the second entails a partial modification of the software of those nodes of the CS that are to be connected to the nodes of another CS. Our aim here is to present a Z80-multimicrocomputer-based structure for autonomous relay systems for computer networking and internetworking.

In the next section, we investigate the reasons for using multimicrocomputer devices to implement relay systems. Since many other multimicrocomputer-based structures have been proposed in the literature, we also discuss our motivations for using the specialized multimicro structure proposed here. We next present the hardware structure of the proposed device, and then describe the basic software that allows processes implemented in a microboard to communicate with processes implemented in the same board or in other ones. Finally, we offer some preliminary measurements on the performance of a device serving as a gateway between a star subnetwork and an X.25 subnetwork.[3]

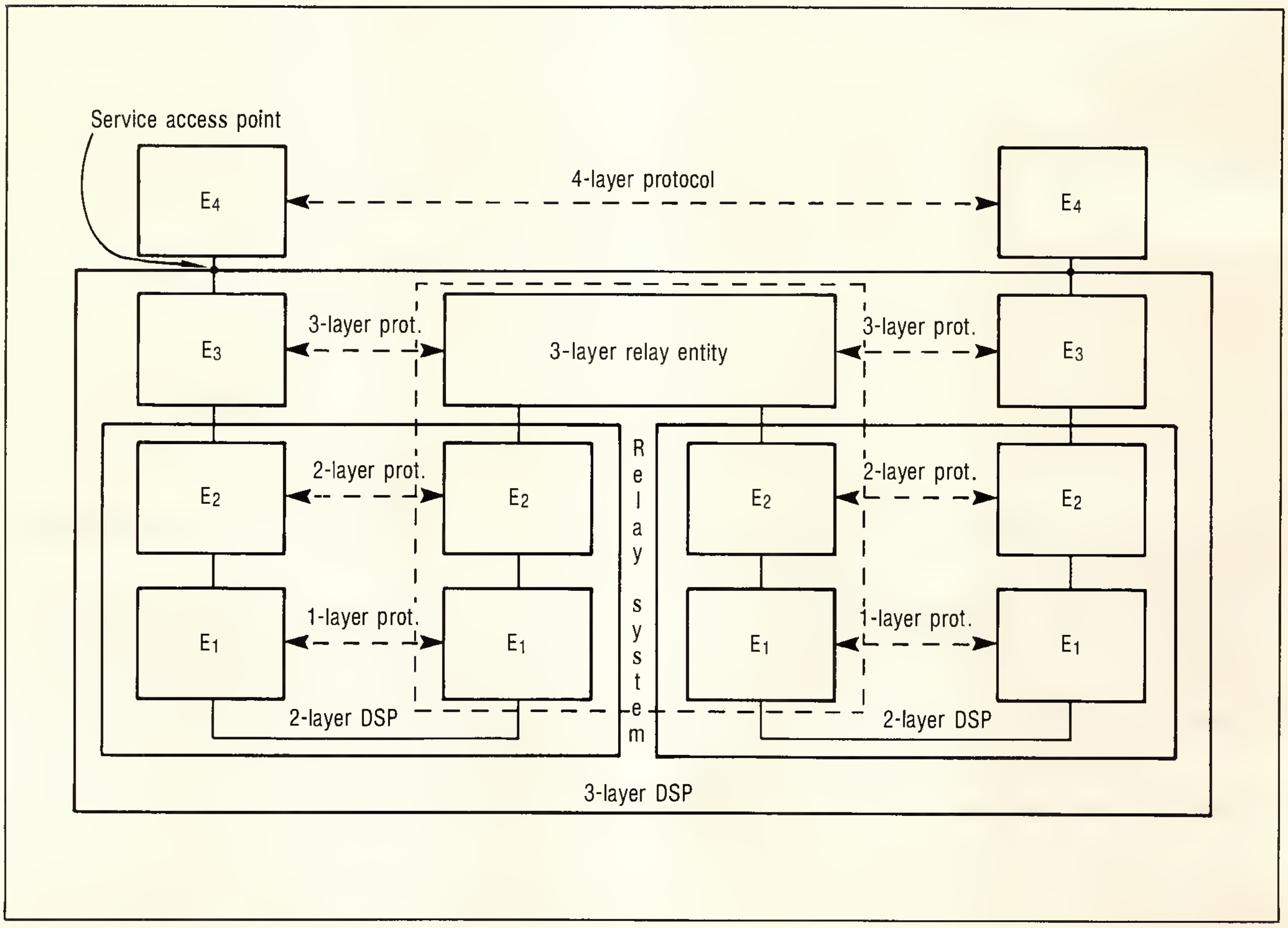


Figure 1. Structure of the distributed service provider (DSP).

Reasons for using multimicrocomputers

One may wish to connect two different CSs through an autonomous relay system for several reasons:

- to save memory space in, and CPU time of, the computer nodes to be connected,
- to avoid having to make significant modifications in the preexisting subnetworks, and
- to easily test and control internetwork data flow.

A dedicated system can be realized as either a minicomputer-based or a microcomputer-based structure. A minicomputer-based solution may be inappropriate in some cases because it does not present a very flexible hardware configuration; for opposite reasons, a solution based on a single microcomputer may be inappropriate. A multimicrocomputer-based structure seems to have the desired flexibility, however. Moreover, because this solution is a *multiboard* one, it produces the desired increase in performance—in terms of throughput and delivery—at minimum cost, since one is required to add only as much hardware as is needed to support the traffic under the desired conditions.

For example, in order to realize a relay system to connect a host computer to an X.25 subnetwork, we would need

- two modules to perform the line and link-level protocols needed to connect the host computer to the relay system,
- three modules to perform the X.25 DTE function needed to connect the relay system to the X.25 subnetwork, and
- one module to perform the appropriate relay function.

Figure 3 shows this arrangement. It is possible to implement these modules either on a single board or on several boards. Software modularity can be provided with either implementation, but only the latter provides the advantages of hardware modularity.

In both implementations, the throughput T of user packets traveling a virtual channel can be deduced by the following formula:

$$T = 1/\max(t_i, i = 1, n)$$

where t_i is the time interval spent within the generic element i of the n elements traveled by a single data unit. Now we derive the throughput in the case of a bidirectional flow through a full-duplex line. In a single-board solution, we have

$$T_l = 1/\max(2n(t_l + t_3 + t_2),\ 8L/V) = 1/t_{lm}$$

where

- L is the data unit length, in bytes, transmitted through the physical line (we note that generally $L = L_c + L_{us}$, where L_c is the length of the control field associated with the text [having an L_{us} byte length] given by the user to the DTE),
- n is the number of network users,
- V is the line rate of bps between the DTE and the node,

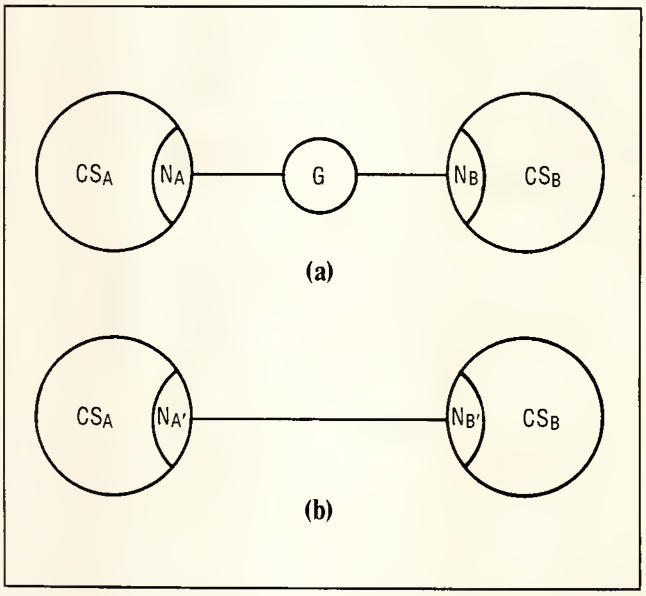


Figure 2. Two ways of connecting two CSs—via a dedicated gateway (a), or directly, by partially modifying the interconnected nodes (b).

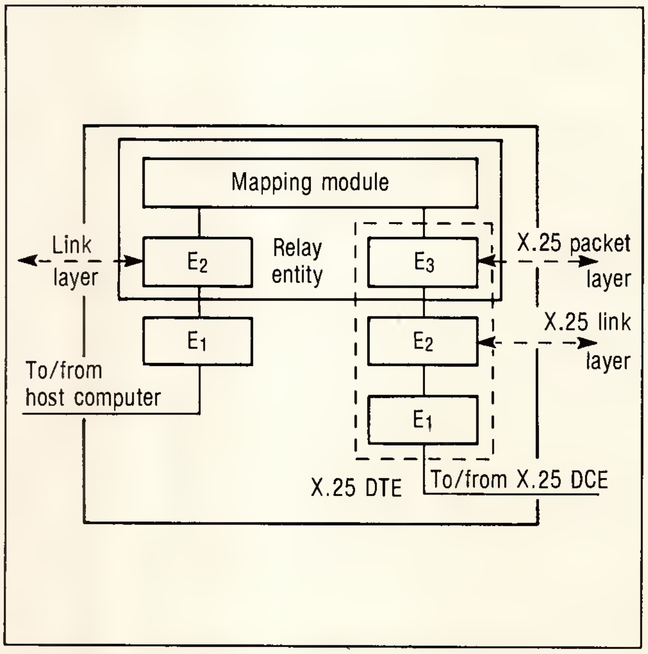


Figure 3. Typical relay system.

- t_2 is the time interval spent inside Level 2 and X.25 Level 1 modules,
- t_3 is the time interval spent inside Level 3 modules, and
- t_I is the time interval spent inside the relay module and the modules performing the protocol between the host computer and the DTE.

In a multimicro solution we have

$$T_2' = 1/\max(2n(t_I + t_3), 2nt_2, 8L/V) = 1/t_{2m'}$$
$$T_2'' = 1/\max(2(t_I + t_3), 2nt_2, 8L/V) = 1/t_{2m''}$$

depending on whether n virtual connections needed by n users are managed by the same board or n different boards.

In teleinformatics, most information passes unchanged through the various layers of a relay system.

It is easy to see that $T_2 > T_1$. This is much more evident when $8L/V$ decreases and/or n increases. In fact, for high values of $t_L = 8L/V$ and $n = 1$, it is easy to obtain an implementation with $(t_I + t_3 + t_2) < t_L$. In this case, the implementation does not necessarily have to be a multimicro solution. For low values of t_L and/or high values of n, however, the positive effect of the parallelism inside the relay system becomes more evident.

For example, in an X.25 DTE working at 9600 bps with $L = 135B$ ($L_{us} = 128B$, $L_c = 7B$), we get $t_L = 112$ ms. The sum of the DTE internal service times can easily exceed this t_L, resulting in a decrease in efficiency that can be avoided by using a multimicro solution. This is even more evident in a DTE for a local-area network. In this case, the network layer is not useful and L_3 must be replaced by L_4 to perform the function of the transport layer. However, the preceding formulas remain unchanged if one substitutes t_4 (the time interval spent by a data unit inside L_4) for t_3. Thus, in an Ethernet-like LAN working at 10M bps and with $L = 1KB$, we get $t_L = 0.8$ ms, and, analogously, in a Proway-like LAN for process control working at 1M bps and with $L = 100B$, we again get $t_L = 0.8$ ms. In both these cases, the parallelism inside the DTE can be used because the sum of the DTE internal service times surely exceeds this t_L value. These results also hold even if we take into account that the packet flow is controlled by a window mechanism to avoid packet loss, duplication, and misordering. In fact, in this case, it is again possible to apply the aforementioned formulas if the window size satisfies the following relation:

$$W_x > (4(t_3 + t_2) + 2(8L/V))/t_x$$

where t_x holds t_{lm}, $t_{2m'}$, or $t_{2m''}$, according to the configuration of the DTE. Thus, to obtain high throughput, it may be necessary to have a large window size and, consequently, a suitably large available memory space.

By defining the efficiency of the system as

$$E_x = \frac{t_L}{t_x} \bullet \frac{L_{us}}{L},$$

we can see that $E_x = L_{us}/L$ for high values of t_L, whereas for low values of t_L this efficiency tends to decrease. In fact, the maximum DTE internal service time exceeds t_L at a certain point, since this internal service time remains practically unchanged when t_L decreases.

We can also see that the delivery delay of the packets can be reduced only by trying to obtain an efficient software/hardware organization—the presence of more boards does not in itself automatically reduce the delivery delay. Our multimicro solution offers the same delivery delay as that of the monomicro solution, but it provides a higher throughput value.

The need for a greater memory space becomes more evident when both the interfaces and the number of channels to be managed inside the relay system increase. In an 8-bit, single-board solution, the maximum memory size is 64K bytes, whereas in a multimicro solution the memory can be increased by adding new boards. For example, if in a multimicro system each board needs 16K bytes of RAM to support information exchange with the other boards, then when a new board is inserted, the total memory will increase by 48K bytes.

Choice of the multimicrocomputer structure

In teleinformatics, most information passes unchanged through the various layers of a relay system. So, for a multimicro solution, it would be useful to have a common memory mechanism to support the exchange of information among the boards.

In such a structure, the crucial question seems to be whether management of the common memory area will be performed by a centralized or a distributed arbiter.[4] A distributed arbiter needs more complicated circuitry but offers higher reliability because even if a board breaks down, the other boards can work normally. However, in our application, when a board goes bad the whole structure breaks down, since each board performs a unique subfunction. Thus, for us, the distributed arbiter offers no greater reliability than the less expensive centralized one. And, even in the distributed approach, a failed board requires that a new board be inserted to reconfigure the system, just as in the centralized approach

the failure of the arbiter or of a board requires that a new arbiter or board be inserted. So, for our application, the choice of a centralized arbiter seemed best.

In designing our multimicro solution, we solved two other problems: the choice of the microprocessor and the choice of the board circuitry. We considered an 8-bit microprocessor suitable for our approach; a 16-bit solution seemed more appropriate for a single-board approach in which the board contains several software modules.

A single-board approach is useful when a computing node has to be strongly integrated with a communication subsystem, or CS, in order to achieve maximum exploitation of CS characteristics (e.g., throughput, reduction of delivery delay, availability). For example, in a robot utilizing a 16-bit multimicro configuration, it can be useful to localize all communication functions in a dedicated 16-bit board in order to obtain a simple virtual machine consisting of several robots strongly interconnected. An application of this kind has been developed by the MODIAC project of the Italian Research Council. MODIAC is a network for process control whose nodes are Z8001-based multimicro devices: one board is devoted to network communication; the others manage specific robot applications.

In certain cases, however, the strong integration of the communication subsystem with network applications is not needed; instead, a high-performance interface for interconnecting a great number of applications to the network is required (e.g., in office automation and distributed data acquisition). In this case, a modular multimicro approach for the communication interface seems to be the most appropriate. Modularity makes it possible to apply the same communication interface, with little adjustment, to a wide range of applications. However, in the computer networking area, a 16-bit module is too powerful for the functions to be performed; in addition, the lower-layer protocols are byte-oriented. Consequently, for our application we considered an 8-bit approach the most appropriate.

Another important problem is board circuitry. We had two alternatives: to use boards already on the market or to design custom boards. For economic reasons, we preferred the latter solution. In order to facilitate board test, we also decided to develop similar boards for all functions in our system. For flexibility, we decided to use very simple kernel circuitry that could be expanded with specialized chips according to user needs. So, by using these strategies, we were able to obtain a low cost in the manufacturing, testing, and use of our boards.

On the market, of course, there were several 8-bit microprocessors that could have been used in our boards. We decided to use the Z80 because of its powerful instruction set, its method of interrupt management (which facilitates the design of our real-time applications), the characteristics of the chips in its family, and especially because of the wide experience of our group in using it.[5,6]

Hardware architecture

For the reasons explained above, we chose a general-purpose, multimicrocomputer-based structure for our relay system for computer networking. It contains several microprocessor units, each making up a complete microcomputer system able to work autonomously and able to supply some resources. In addition, our system was designed in a modular way so that the number of microprocessor units, and some of the internal resources, could be changed without having to change the remaining parts of the system. Each unit can communicate with other units through a common memory, or with external systems through suitable interfaces implementing a standard RS-232, or synchronous HDLC, protocol. The architecture we adopted is shown in Figure 4; some microprocessor units can access common resources on a memory board through a common bus.

We chose a general-purpose, multimicrocomputer-based structure for our relay system.

We made a great effort to design the basic structure of our microsystem so that it would be appropriate for a wide range of computer networking applications. The basic hardware mechanisms and software primitives can be enhanced according to the requirements of the particular application. We will discuss basic software primitives later in this article; the section below describes the hardware primitives for common bus access, and the information exchange inside the microsystem.

Information exchange. The information flow inside the system can be of two kinds:

- information exchange between two microprocessor units, and
- information transfer between a microprocessor unit and a common memory board.

Both of these information flows are handled by the common bus. Only a microprocessor unit can obtain access to the common bus. To avoid conflicts, access to the common bus is managed by a bus controller.

Requests for common bus access. There are two kinds of bus access request—the normal request and the extraordinary request.

Normal request. This request is handled in a software-transparent way and hence can be performed very quick-

ly. It is activated every time a microprocessor unit wants to perform a read/write operation in the common memory or an information exchange with another microprocessor unit. After a single read or write operation, the microprocessor unit releases the common bus. Any conflicts which occur when bus access is requested are managed by the bus controller.

Extraordinary request. This request is activated by the software and allows a microprocessor unit to hold the common bus for the whole instruction sequence. The need for such a request arises from the existence, in many applications, of common variables (e.g., semaphore variables) which can be handled by any microprocessor unit (but only by one microprocessor unit at a time). For this reason, the routine that manages such variables must, first of all, activate an extraordinary request line and, at the end, reset this line to inform the bus controller that the common bus is available to another user.

It is useful to work under a disabled interrupt when a unit is performing an extraordinary request, since this shortens the time during which the bus is busy. But when more than one extraordinary request is performed, a station could wait too long, possibly creating dangerous situations if some routines need interrupts to work (as in HDLC, for example). For this reason, each board is provided with internal logic that can generate a nonmaskable interrupt (NMI) when several extraordinary requests are performed. In this situation, the BUSY line is driven low, which the bus controller detects. In this way, a routine to detect and manage interrupts is activated.

Let us now examine the two chief components of our system—the bus controller and the microprocessor unit board.

Bus controller. This unit manages the access to the common bus. When it receives several bus request signals, it acknowledges only one according to a fixed priority—by means of the ACK BUS signal—and puts the others in a queue. Such operations are performed in a parallel fashion.

The functions performed by the bus controller can be described by means of a synchronous sequential machine model (Figure 5). Bus requests (BUS 1, BUS 2..., BUS*n*)

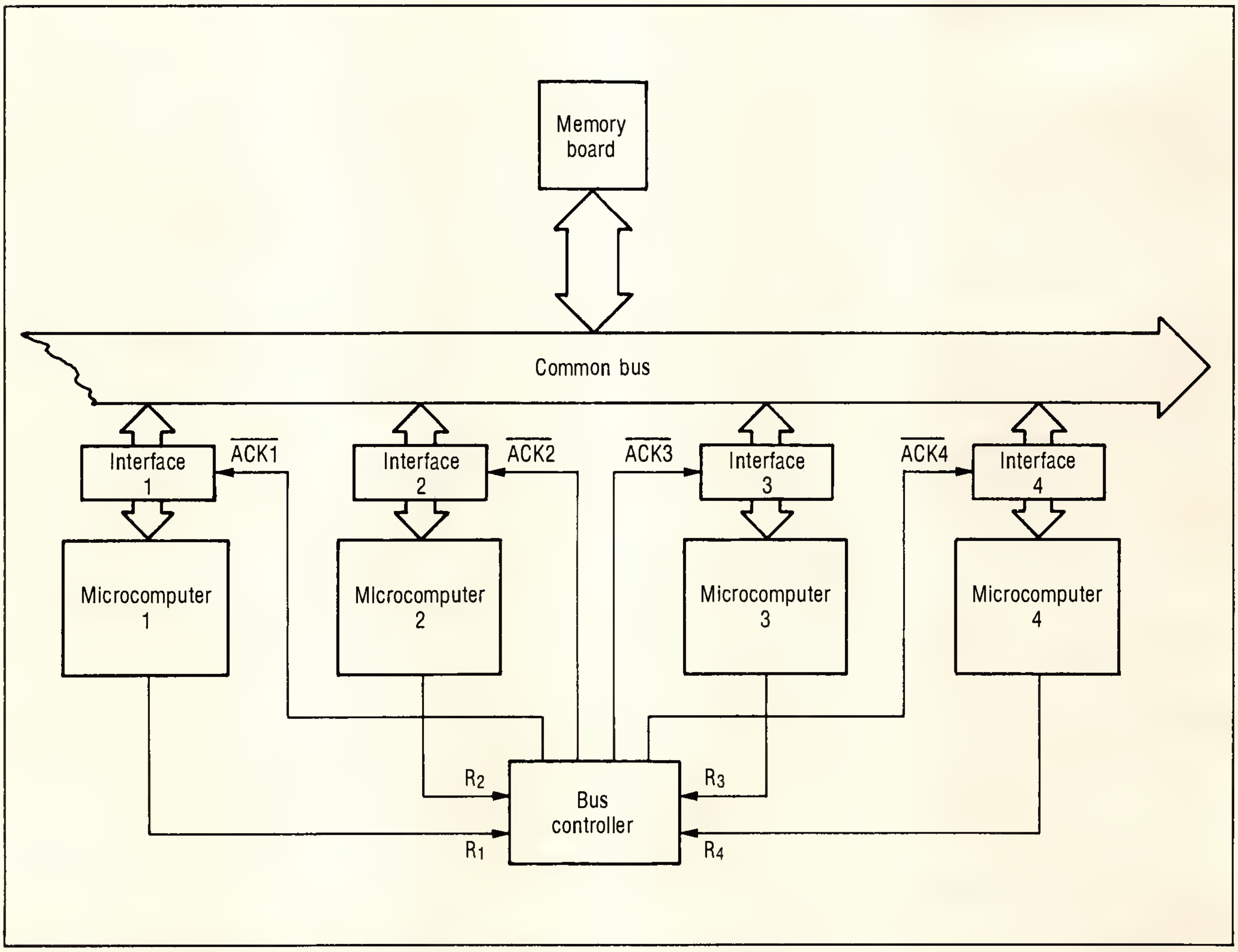


Figure 4. The multimicrocomputer architecture adopted for our project.

coming from the microprocessor units are input variables, while the signals from the bus controller to the microprocessor units are output variables. The latter variables are

- $\overline{\text{EXT WAIT}i}$, which, when it is active, signals to microprocessor unit i that the common bus is busy, and
- $\overline{\text{ACK}i}$, which, when it is active, signals to microprocessor unit i that it can have access to the common bus.

There is also a $\overline{\text{BUSY}}$ line, which signals the bus controller that a microprocessor unit is performing an extraordinary access to the common bus. This prevents the bus controller from transferring the bus to another microprocessor unit until after the current instruction has been completed. Finally, there is the $\overline{\text{BM1}}$ signal, which is used as a clock by the bus controller to synchronize its internal logic transitions.

Depending on the input variable configuration, the bus controller produces an appropriate output according to the following rules:

- To avoid conflicts on the bus, only one $\overline{\text{ACK}i}$ line at a time can be active.
- When an $\overline{\text{ACK}i}$ line is active, all $\overline{\text{EXT WAIT}j}$ ($j=1,2,\ldots,n$; $j \neq i$) lines must be active.

A microprocessor unit disregards these signals if it does not need to perform a common bus request.

All common bus requests are acknowledged according to a priority that is assigned in a static or dynamic way. We have taken a dynamic approach and have adopted a cyclic priority scheme in which each unit has a priority whose value shifts after each common bus request.

Figure 6 shows a simplified diagram of the bus controller. Input/output relationships are handled by an EPROM. Signals $\overline{\text{D BUS1}}$ to $\overline{\text{D BUS}n}$ supply the address to the EPROM, while signals $\overline{\text{ACK}i}$ ($i=1,\ldots,n$) and $\overline{\text{EXT WAIT}i}$ ($i=1,\ldots,n$) are outputs. One can easily program the EPROM to set priorities among various bus requests. When each unit has a fixed priority, it is possible to use a binary counter to sequentially shift the selected memory blocks.

Microprocessor unit board. All the microprocessor units have been designed in the same way so as to make the architecture modular. A board is adapted to a particular function either through software or through elimination of those chips which are not needed for the application.

Figure 7 shows a block diagram of our microprocessor unit. There are three major sections:

- a central section, containing unit resources,
- an interface to the common bus, and
- an interface to external systems.

Central section. This section consists of a Z80 CPU, a 12K-byte memory block (8K bytes of EPROM and 4K bytes of static RAM), two I/O devices (a Z80 SIO and a Z80 CTC), memory decoder logic, and I/O decoder logic.

The memory decoder logic produces the signals that enable the RAM and ROM memory and the $\overline{\text{EXT MEM}}$ signals to perform a hardware request for access to the common bus. A suitable decoding of the A_{13}, A_{14}, and A_{15} addresses is obtained through the use of the following memory map:

0 : 8K bytes EPROM,
8 : 32K bytes internal RAM, and
32 : 64K bytes external RAM (on a common memory board).

The I/O decoder logic produces the signals that enable the CTC and SIO, as well as

- $\overline{\text{BRQ}}$, extraordinary request for access to the common bus,
- $\overline{\text{RES BUSY}}$, end of extraordinary access to the common bus, and
- $\overline{\text{EXT I/O REQ}}$, request for access to the communication interface block of another microprocessor unit.

Interface to the common bus. This interface is made up of three blocks: common bus request logic, a communication interface block, and a three-state interface.

The common bus request logic produces the signals needed to request access to the common bus, receives con-

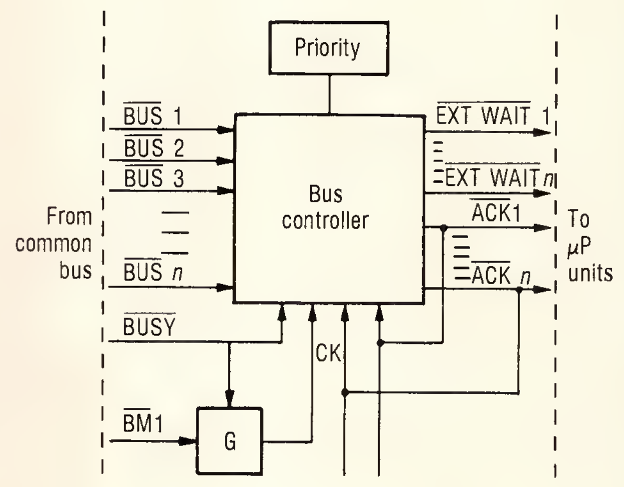


Figure 5. Synchronous sequential machine model of the common bus controller.

trol signals generated by the bus controller, and drives three-state buffers to obtain the access to the common bus. BUS*i* are the input signals from the bus controller; WAIT and NMI are input signals generated by the common bús request logic's own CPU.

The communication interface block, or CIB, makes direct communication possible between two microprocessor units or between a microprocessor unit and itself. This interface has been implemented as a Z80-compatible I/O device board, and it is used by other microprocessor units as a logic gate to activate, via an interrupt, the receiving processes implemented either in another board or on the same board. Each CIB can be addressed by a common bus ordinary request (EXT/ORQ) activated by the output instruction

OUT (CIB address) A

where "CIB address" is the address of the CIB belonging to the destination microprocessor unit, and A is the accumulator to be set equal to 1.

The three-state interface is needed to physically connect each microprocessor unit to the common bus. It contains several three-state buffers that are usually disabled, except when they are enabled by the bus controller to allow a common bus access.

Interface with external systems. This RS-232 interface makes it possible to connect external devices such as video terminals or teletypewriters to the multimicro system. Data are serialized by the Z80 SIO—channel A is used as a reception gate and channel B as a transmission gate. The data rate is fixed by the Z80 CTC.

Basic software architecture

Each microprocessor unit contains a group of routines constituting the kernel of an operating system that supplies some primitives, at different levels, for the processes located in the system. This kernel was conceived with our teleinformatics applications in mind.

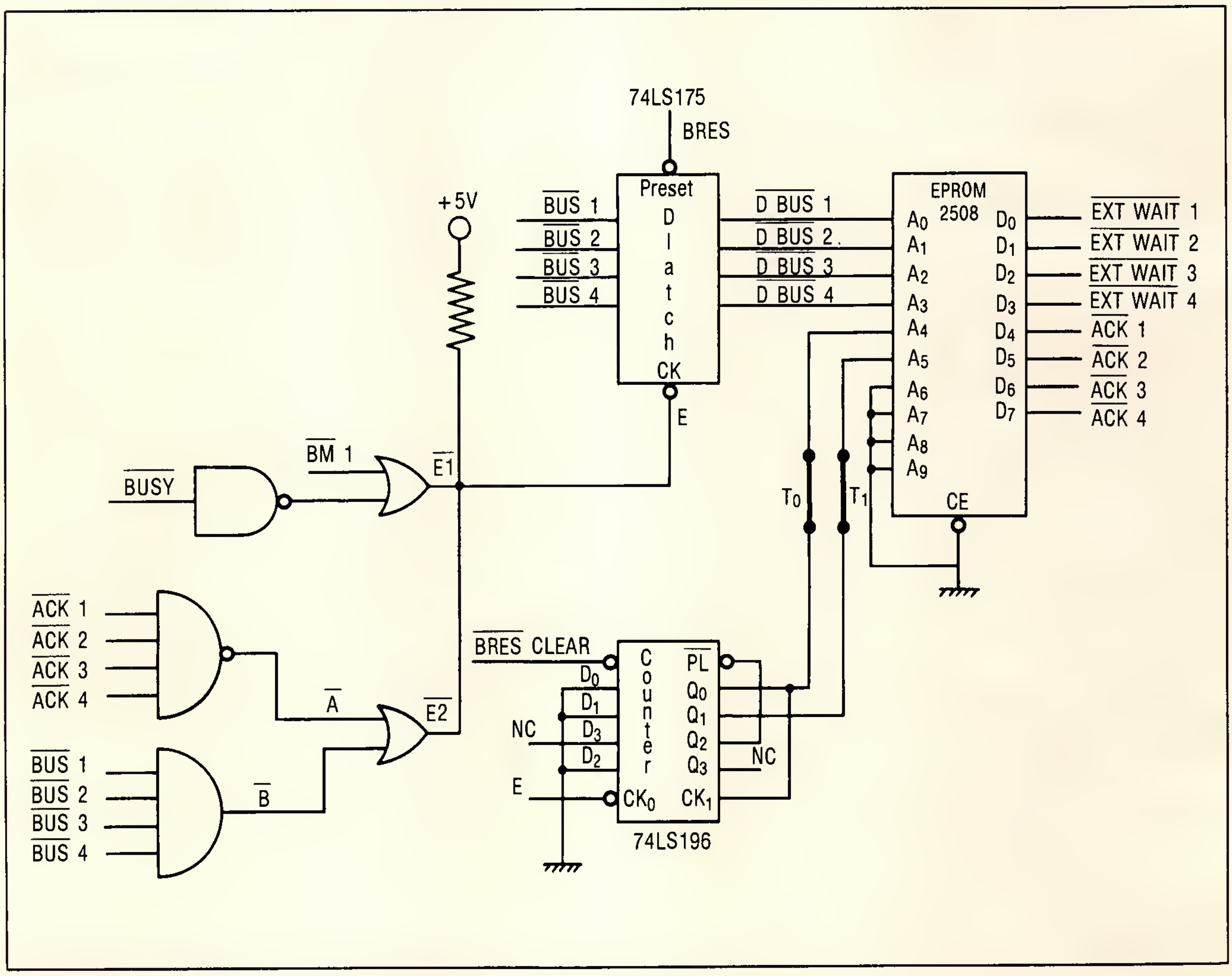


Figure 6. Simplified hardware diagram of the common bus controller.

The primitives can be classified as follows:

- primitives for message exchange,
- primitives for critical region management,
- primitives for resource and time-out management, and
- primitives for process management.

All the service routines are PLZ/SYS-consistent, and generally they present two entry points: one for ASM calls and the other for PLZ calls.

Primitives for message exchange. Message exchange among microcomputer units is performed by SEND/RECEIVE software primitives. They are at the lowest level of the basic operating system and use the BUS REQ hardware mechanism to obtain access to the common memory area; $n*(n-1)$ mail boxes (where n is the total

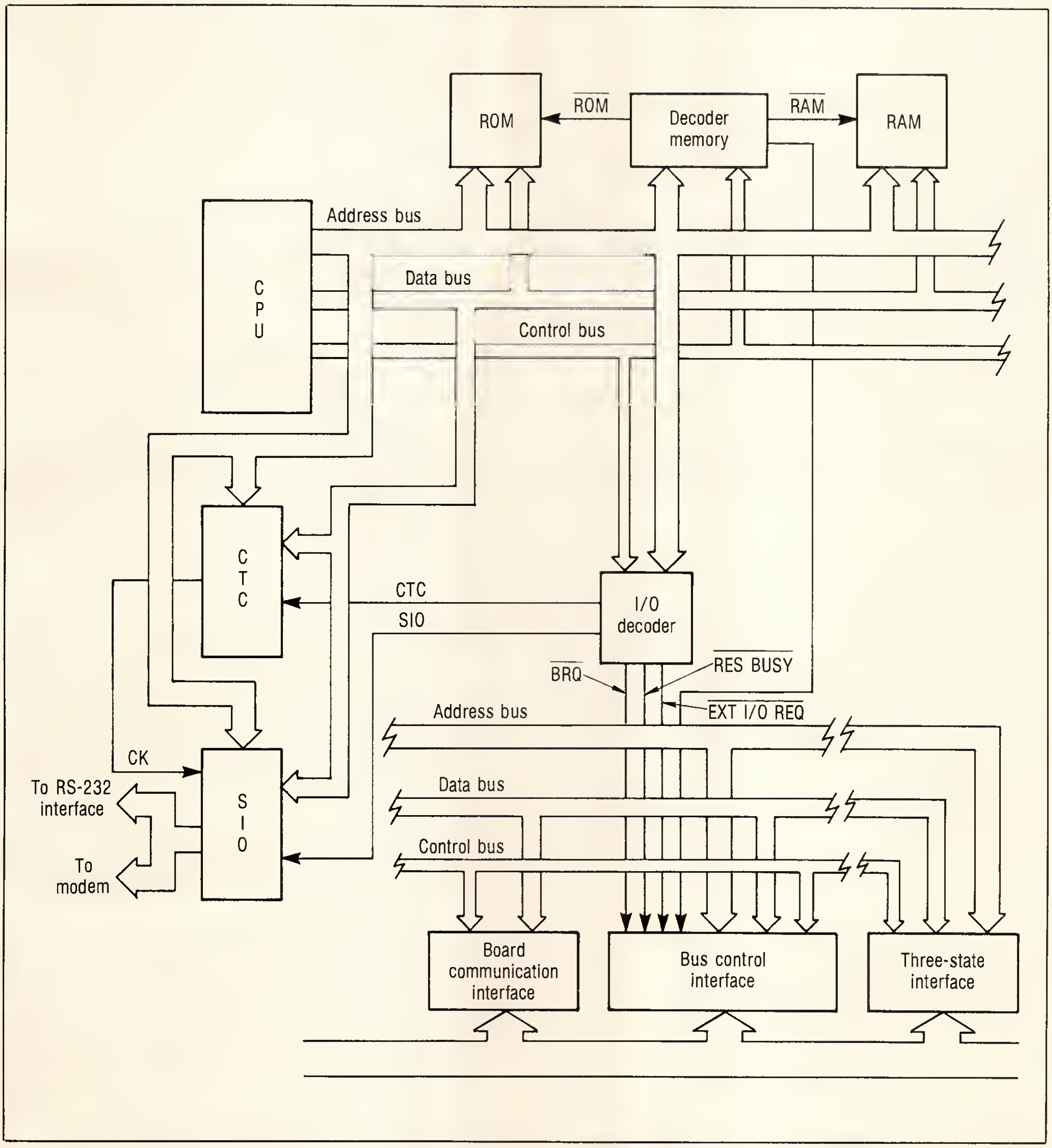


Figure 7. Block diagram of the microprocessor unit board.

number of micro units) are placed in the common memory board. The structure of the mailboxes is defined as follows:

TYPE
MAILBOXES array ($n*(n-1)$ word)

Into (out of) this mailbox, the SEND (RECEIVE) primitive of the source (destination) unit deposits (removes) a two-byte length message to be transmitted to (received from) the destination (the source unit).

Two types of messages can be exchanged by the following structure:

COMMAND INFORMATION (CMD-INF)
BUFFER INFORMATION (BUFF-INF)

CMD-INF consists of an operating code (one byte long) used to start a process. BUFF-INF consists of the address (two bytes long) of a buffer located in the common memory board. Memory mapping eliminates confusion between "command information" and "buffer information."

Use of the SEND primitive. The user processes see the SEND primitives as procedures defined as

SEND procedure (MESSAGE word, D_UNIT byte),

where MESSAGE is the two-byte information to be exchanged between two units, and D_UNIT is the destination unit.

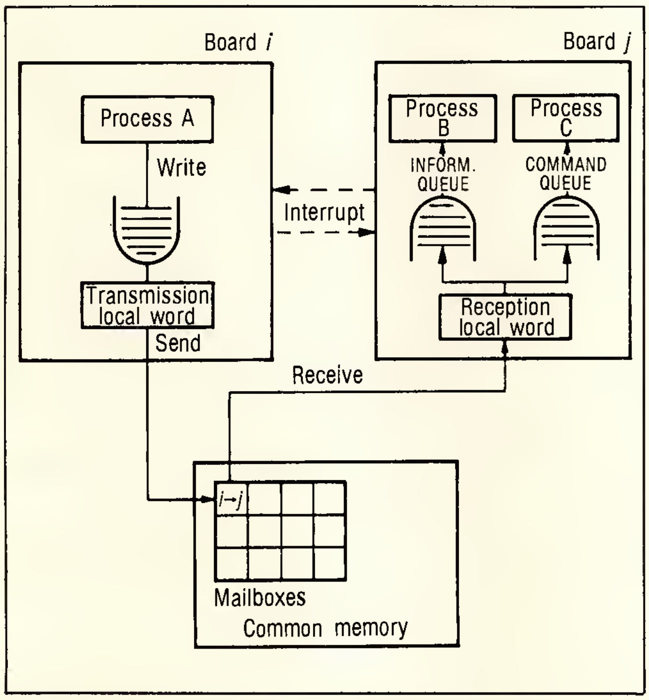


Figure 8. SEND/RECEIVE procedures exchanging a message via the common memory.

The SEND procedure inserts the MESSAGE in the output queue of the source unit associated with the D_UNIT (Figure 8). It then activates a local sending process that operates on this queue to transfer the MESSAGE to the appropriate mailbox. This local sending process uses two semaphores to avoid a loss of information during the transfer of this MESSAGE from the local to the common board:

- The first semaphore is related to a "transmission local word"; it is reset only after the MESSAGE has been transferred to the mailbox.
- The second semaphore is related to the mailbox itself; it is reset only after the RECEIVE function of the called unit has performed the get operation.

The sending process consists of two subprocesses:

- The first subprocess tries to transfer the MESSAGE from the queue to the transmission local word by using the first semaphore. It activates the second subprocess by means of an interrupt (via the CIB). It remains in the ready state until the queue is not empty.
- The second subprocess tries to transfer the MESSAGE from the transmission local word to the appropriate mailbox in the common memory using the second semaphore. It enables the first semaphore in the case of a successful transfer. In any case, it activates (through the CIB) the RECEIVE primitive of the called unit and then ends.

The fast transmission of data is performed by the following primitive:

F_SEND procedure (MESSAGE word, D_UNIT byte)

After the transfer of the current MESSAGE from the transmission local word to the mailbox, this primitive puts its own message into the transmission local word and activates, by an interrupt via the CIB, the subprocess that transfers MESSAGES from the local word to the appropriate mailbox.

Use of the RECEIVE primitive. The RECEIVE primitive is activated by an interrupt within each board. This interrupt is generated by the device CIB, which is enabled by the calling board.

The interrupt method is the IM2 Z80; consequently, the device CIB will provide, in the interrupt vector, the information needed to identify the calling unit. Therefore, the RECEIVE function will offer several entry points, with each determined by the value of the interrupt vector. This fosters identification of active mailboxes.

Once it has been verified by a local semaphore that the last MESSAGE has been correctly received, the mailbox value is transferred, by the RECEIVE primitive, into a "reception local word." Then the mailbox is reset to

enable any additional sending. An interrupt is also sent to the sending process of the calling board in order to force the transfer of another MESSAGE, if any, from the transmission local word of the calling board to the appropriate mailbox.

The MESSAGE is handled in one of two ways, depending on whether it is a CMD-INF or a BUFF-INF. BUFF-INF is inserted in an input queue, whereas for CMD-INF a special procedure serving the commands is activated.

Critical region management. Critical regions, which are located in the common memory board, are characterized by the fact that a process can have access to them only after the previous process has completed its task. For this reason, every time a process needs to work in a critical area, it can behave in one of two ways:

- It can reserve the bus for the entire task sequence to be performed by an extraordinary request. However, this prevents all other users from obtaining access to the common memory.
- It can use semaphores located in the common memory to control access to the critical regions. A SPACE-REQ (S_i) primitive tests semaphores S_i, performing a test-and-set operation via an extraordinary bus request.

The first method penalizes all microprocessor units that permit only one unit to access the common memory for a long time. The second one avoids this situation because an extraordinary bus request is performed only for the short time needed for testing and setting a semaphore. But if concurrent processes are in the same microprocessor unit, it is possible that they may wait for the same resource. It is then necessary to implement a supervisory function to put those processes not needing resources in a waiting space, and to put the first ready processes in the ready queue in a running state.

Memory management. As we have already said, it is possible to exchange long buffers by means of the mailbox structure, indicating the addresses of those buffers available in the common memory area. Such buffers therefore must have a standard format in order to make available to any process all the information belonging to a buffer.

To satisfy this request, a free queue of buffers has been created in the memory board; each element in this queue has the structure shown in Figure 9. So that such buffers can be managed with PLZ/SYS, they have been defined as

```
TYPE
BUFFER RECORD (OFFSET RECORD (LINK ↑
               BUFFER, INDX, STATUS,
               LENGTH byte), DATA__UNIT
               array (150 bytes))
```

where

- LINK is a pointer to another BUFFER,
- INDX is the index of the DATA__UNIT array from which the useful information starts,
- STATUS is used to link several buffers in order to obtain a data unit having a length greater than 150 bytes (in addition, STATUS is used by HDLC to establish whether the buffer has to be put in the free queue), and
- LENGTH is the length of the useful information, starting from the index INDX, contained in the DATA__UNIT array.

Note that the choice of the byte length of a DATA__UNIT depends on the maximum length of the frames traveling through the communication subsystem. For example, the aforementioned 150-byte length is selected for an X.25 communication subsystem.

All the above buffers are linked, by the link field, in a queue whose elements are available to all the units working inside the system. The descriptor of this queue is defined as

```
TYPE
QUEUE TYPE RECORD (FIRST, LAST↑BUFFER,
                   CARDINAL word)
```

where FIRST, LAST are pointers to the first and the last element in the queue, and CARDINAL is the number of the queued elements. This descriptor is placed inside a critical region, so it must be managed according to the above rules.

A number of routines are provided as primitives to carry out I/O in the queues. These primitives are offered at two levels:

```
(1)        CALL PUT__FREE
           CALL GET__FREE
```

These routines are written in ASMZ80, and suitable registers must be used to exchange the buffer address and

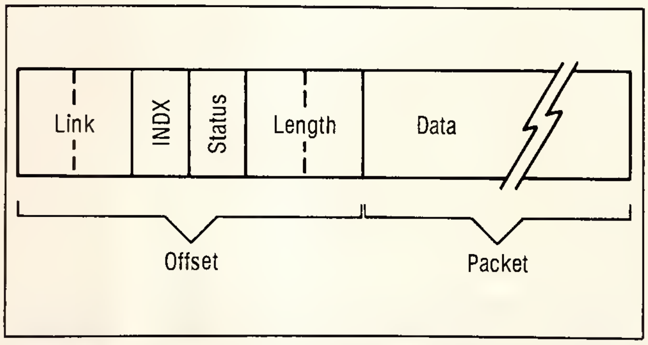


Figure 9. Structure of each element in the free queue of buffers in common memory.

to determine if the requested operation has been successful or not.

(2) PUT__FREE__BUFFER procedure (ADDRESS↑ BUFFER)
returns (RESULT byte)
GET__FREE__BUFFER procedure
returns (ADDRESS↑ BUFFER, RESULT byte)

These routines are written in PLZ, and RESULT indicates if the requested operation has been successful or not.

All the above primitives execute the I/O requests after having performed a test-and-set operation on a semaphore associated with the free queue descriptor. The I/O requests on the queue can be usefully executed under supervisor control. In this case, the supervisor puts the process into a waiting state if the access to the critical region is impossible.

Primitives to manage the I/O of any queue are also available; their descriptors are similar to those related to the free queue.

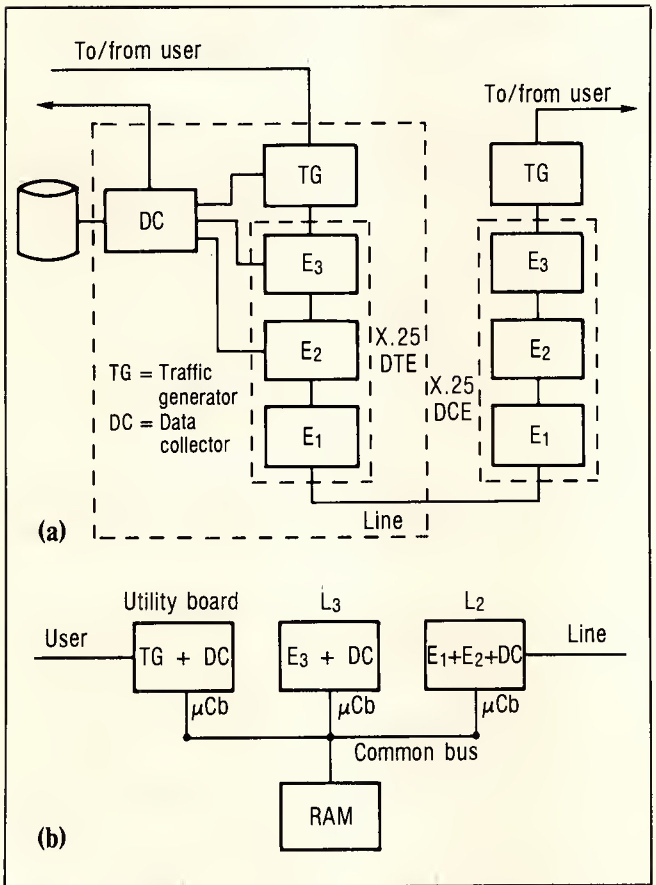


Figure 10. Functional structure of an X.25 DTE/DCE interface for measurement collection (a); hardware structure of an X.25 DTE (or DCE) with a traffic generator and data collector (b).

Time-out management. To manage periodical events in the system, some service primitives for time-out management have been provided through the Z80 CTC chip available on each unit. These primitives allow time-outs to be dynamically changed, stopped, and restarted. The time-out management structure is based on a suitable array, defined as

COUNTER array (*n* word)
START COUNTER array (*n* word)
EVENT ADDRESS array (*n* word)

where *n* is the total number of time-outs. The corresponding elements of these arrays are referred to the same time-out. In particular,

- COUNTER contains the number that must be decreased every *m* microseconds by a routine activated by the CTC via interrupts (*m* is fixed during the initialization phase of the CTC),
- START COUNTER contains the starting value of the time-out (if this value is zero, the time-out is not activated), and
- EVENT ADDRESS contains the address of the event to be set when the time-out fires, that is, when the counter becomes equal to zero.

The primitives for time-out management are

START__TIMEOUT procedure (TIMEOUT__ NUMBER byte, TIMEOUT__ VALUE word)
STOP__TIMEOUT procedure (TIMEOUT__ NUMBER byte)

Process management. When we presented the above primitives, we referred to a supervisor able to optimize the scheduling of processes concurrent to the same resource. If all the primitives are utilized directly by the processes, the supervisor is only an event scheduler, with each event associated with a process. The process can be only in one of two states:

- inactive, when the process does not need the CPU, or
- active, when the process needs the CPU.

An active process can be either in the ready state or in the running state. As a consequence, the continuous test-and-set operation on a semaphore that may be performed by an active process can leave that process in a useless running state until the resouce is obtained. This situation stops all other processes which are in an active-ready state and also produces continuous, useless requests for extraordinary bus access, the effects of which are felt by the whole system.

If, however, we keep in mind that the processes obtain the services offered by the primitives through other (higher-level) primitives addressed and controlled by the supervisor, we can optimize the access to the common memory and to the other resources. Of course, we can improve the supervisor by adding new functions to obtain a more powerful distributed operating system. However, this would not be practical for our applications.

An example

Our Z80 multimicro structure was used as a stand-alone X.25 DTE device directly connected to a similar device working as an X.25 DCE (data circuit equipment) to collect measurements (Figure 10).[3] Both the X.25 DTE and the X.25 DCE consist of the following boards:

- Board L2 (see again Figure 10) manages the line and the link protocols. The link protocol consists of two parts: a hardware part performed in the Z80 SIO that works according to the HDLC protocol, and a software part that works according to the X.25 recommendation. The software part opens and clears a link between the DTE and the adjacent DCE. In addition, it performs flow control and recovery on the data (called frames) traveling over this link.
- L3 manages the network protocol. It opens and clears virtual connections through the X.25 communication subsystem. In addition, it performs flow control and recovery on the data (called packets) traveling over these virtual connections.
- A utility board generates an artificial traffic of packets in order to collect measurements.
- A common memory board is shared by L2, L3, and the utility board.

To evaluate the throughput of this X.25 connection, we produced an artificial packet flow from the DTE to the DCE, and vice versa. In each test, we fixed the packet generation frequency, the packet length, the packet contents, and the line bit rate. The test results (Figures 11 and 12) led us to several conclusions:

- When the line rate increases, the throughput also increases, but the system efficiency decreases. This is because the elaboration time becomes more significant as the line transmission rate increases. However, the efficiency remains high even when the line rate increases.
- The efficiency and throughput decrease when the packet length decreases. When the packets have shorter lengths, less useful information is sent during a given service time.

Similar measurements were performed on a dedicated 8080-based multimicro DTE,[7] and on a software DTE

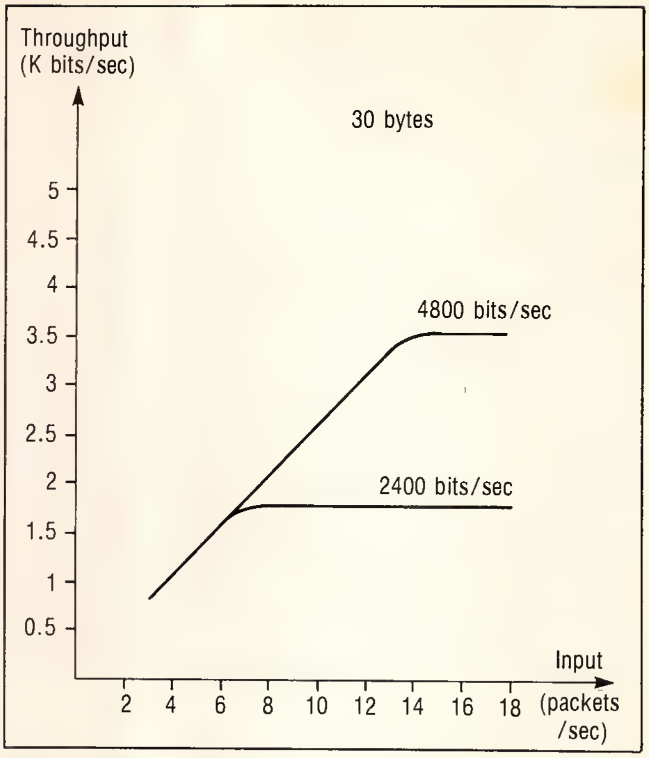


Figure 11. Total throughput (in K bits/sec) vs. applied load (in packets/sec), with a 30-byte packet length and a window size at the link layer equal to 3.

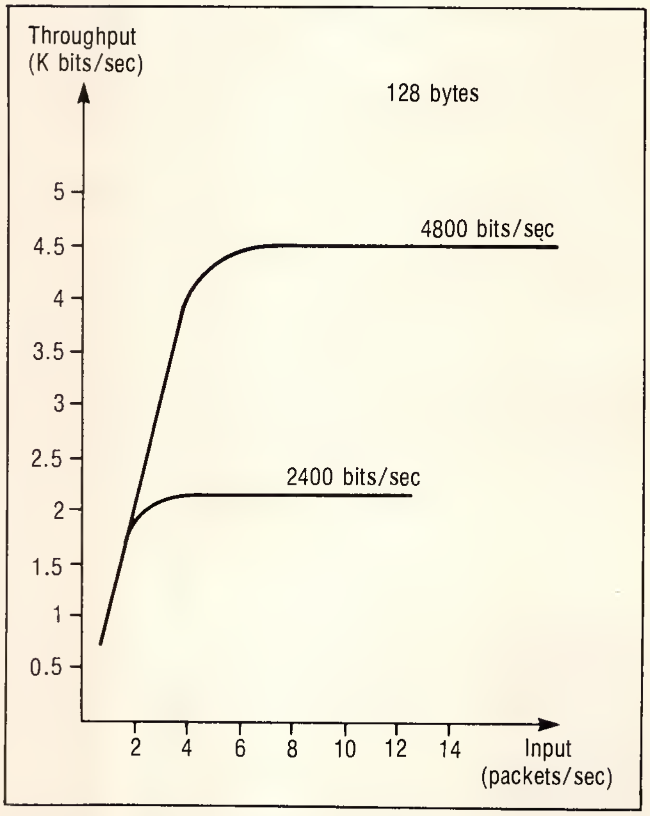


Figure 12. Total throughput (in K bits/sec) vs. applied load (in packets/sec), with a 128-byte packet length and window size at the link layer equal to 3.

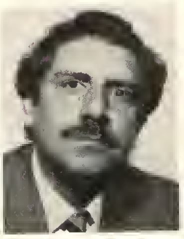
implemented on a Univac 1100/81.[8] These measurements show that the maximum efficiency obtainable at the network layer (for a 2400-bps line rate, with L_{us} = 128B, n = 1 user, and text contents = zero) is 94.8 percent for the 8080-based DTE (at W = 7) and 90 percent for the software DTE.

The efficiency of our DTE is 94.8 percent (at W = 3) under the same conditions. Since L_{us}/L = 128/135 = 94.8 percent, we derive from Equation 2 above that $t_x = t_L$ in both the 8080-based DTE and our DTE. This means that t_2 and t_3 are less than t_L (recall that in this case $t_I = 0$ because the traffic generator is internal to the DTE). However, our DTE reaches this maximum value at W = 3, whereas the 8080-based DTE reaches it at W = 7. So we derive from Equation 1 that the sum of the service times in our DTE is less than that of the 8080-based DTE; thus, in our DTE a smaller memory space is needed to obtain the same throughput. This also means that our DTE produces a shorter delivery delay than that produced by the 8080-based DTE. On the Univac, the efficiency of the DTE implemented shows that the sum of the internal service times exceeds t_L and that it is not possible to reach our DTE's maximum efficiency by increasing the window size. These considerations show that a well-integrated software/hardware architecture can contribute to very high performances.

Our multimicro device can serve as a dedicated system for a wide range of computer networking applications. We made a great effort to simplify and integrate the hardware and software architectures so as to obtain maximum performance at minimum cost. A comparison with similar devices gives very satisfactory results. At the present, we are testing our device for use as a certified DTE for Itapac, the X.25 Italian public network. We are also developing it to serve as a gateway between Itapac and typical LANs such as Ethernet and Proway. And we are studying improvements designed to provide higher performances for applications which need very high transmission line rates. ■

References

1. "OSI Reference Model," ISO TC97/SC16, N.7498, 1982.
2. A. Faro and G. Messina, "Internetworking Analysis," *Computer Communications J.*, No. 8, 1982.
3. "X.25 Recommendation" ("Orange Book"), CCITT, 1979.
4. B. A. Bowen and R. J. A. Buhr, *The Logical Design of Multiple-Microprocessor Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1980.
5. P. Cerami, A. Faro, and O. Mirabella, "Communication Aspects in Microcomputer Environments," *Proc. ICCC 81,* New York, 1981.
6. A. Faro, G. Messina, and L. Vita, "A Microgateway for Computer Networks," *Proc. MIMI 82,* Paris, 1981.
7. G. Andreoni, G. Kacin, and D. L. A. Barber, "Some Initial Measurements of Euronet Performances Using EIN Microprocessor Units," *Proc. ICCC 81,* Atlanta, 1981.
8. A. Faro, G. Messina, and F. Martinucci, "On the Measured Behaviour of a X.25 Subnetwork," *ACM Performance Evaluation Review,* Special Issue on Sigmetrics 83, Aug. 1983.



Alberto Faro is an associate professor of computer science on the engineering faculty of the University of Catania, Italy. He is also director of the university's Informatics and Telecommunications Institute and of its computer center. Active in the ISO/OSI and IEC working groups on distributed systems standardization, he is interested in the architecture of, and formal description techniques for, distributed systems.

Faro received the engineering degree from the Politecnico di Milano in 1971.

Orazio Mirabella is a member of the engineering faculty of the University of Catania, where he performs research on distributed system design under the Computer Science Program of the Italian National Research Council. He is active in the IEC working group on distributed systems standardization. His research interests include multimicroprocessor system design and LAN protocol performance evaluation.

Mirabella received the physics degree from the University of Catania in 1971.

Lorenzo Vita is a researcher in computer science on the engineering faculty of the University of Catania. His research activity is also supported through the Computer Science Program of the Italian National Research Council. A participant in the Esprit Project of the European Community to define and implement software tools for protocol design, he is interested in the specification and performance evaluation of distributed systems.

Vita received the engineering degree from the University of Catania in 1979.

Questions about this article can be directed to the authors at the Istituto di Informatica e Telecomunicazioni, Facoltà di Ingegneria, Università di Catania, Viale A. Doria 6, 95125 Catania, Italy.

A cyclic redundancy code, or CRC, is often used to ensure the integrity of messages in data communications. This program for generating the CRC is faster than its bit-wise counterpart.

"On the Fly" CRC-16 Byte-wise Calculation for 8088-based Computers

D. V. Shouse

General Railway Signal Company

A cyclic redundancy code, or CRC, is commonly employed to ensure the integrity of messages in data communications. In a system that uses a CRC, the message being transmitted is considered a binary polynomial. This polynomial is used with an arbitrary generator polynomial in a message-error-detection algorithm. The CRC-16 generator polynomial is a standard used throughout the world.

The CRC-16 calculation is needed in a number of computers. It can be obtained by means of hardware or software. Software calculation of the CRC-16 value can be done in a number of computer languages. For example, Aram Perez, in the June 1983 issue of *IEEE Micro,*[1] presented a byte-wise method for calculating the CRC-16 result that used Intel 8085 assembly language.

This article shows how "on the fly" CRC-16 calculation can be performed in Intel Fortran on an Intel-8088-based computer, using Perez' method. The routine is assembled by Intel's Fortran-88 assembler, which in turn produces Intel-8088-generated code. The resulting program is faster than a bit-wise CRC-16 program, but the number of bytes it requires is greater. However, careful examination of the code generated by Intel Fortran shows that one can reduce the number of bytes by recoding the program in Intel-86 assembly language. For example, in Intel Fortran the exclusive-OR function is implemented with AND-OR logic, whereas in Intel-86 assembly language it can be done with an exclusive-OR instruction. Moreover, in Intel-86 assembly language there are individual instructions to test parity and carry flags, whereas in Fortran the test for these conditions requires several instructions. These differences alone result in a 106-byte reduction in the size of the code.

How does one use the CRC-16 routine for verifying transmission and reception of messages? Assume that the message to be transmitted consists of x eight-bit bytes. The total length of the transmitted message, then, is $x+2$ eight-bit bytes. The extra two bytes are the CRC-16 value on the first x bytes. The receiving logic then does a CRC-16 calculation on $x+2$ bytes. If the resulting

0272-1732/85/0400-0067$01.00 © 1985 IEEE

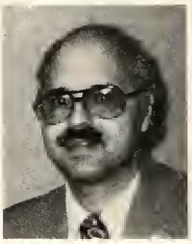
CRC-16 value is zero when all the bytes of the message have been received, the message is valid.

The routine must be modified if it is to be used on an IBM PC, since that machine does not support some of the special Intel instructions.

Both the code generated from Intel Fortran and the ANSI-standard Fortran code are given below so that the reader may compare them. Note that the latter requires 1176 bytes whereas the Intel code needs only 391. ■

Reference

1. Aram Perez, "Byte-wise CRC Calculations," *IEEE Micro*, Vol. 3, No. 3, June 1983, pp. 40-50.



D. V. Shouse is a senior design engineer with the General Railway Signal Company in Rochester, New York, where he is in charge of designing hardware and software for microprocessor-based centralized railway traffic control systems. He teaches a microprocessor course at the Rochester Institute of Technology. Before joining General Railway Signal in 1972, he worked for several companies in the areas of hardware and software system design.

Shouse received a BSEE degree from Bradley University, Peoria, Illinois, in 1966, and an MSEE from the Rochester Institute of Technology.

Questions about this article can be directed to Shouse at the General Railway Signal Company, PO Box 600, Rochester, NY 14692.

Intel Fortran listing

```
FORTRAN-86 COMPILER
CRC16.F88

SERIES-III FORTRAN-86 COMPILER V2.1
COMPILER INVOKED BY: FORT86.86   CRC16.F88

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       SUBROUTINE CRC16 — CALCULATE CRC16 VALUE
C       INTEL ROUTINE
C
C       CALLING SEQUENCE:        CALL CRC16 ( DATA BYTE, CRC VALUE)
C
C       CALLED BY:   CLVRTC
C       CALLS:       NONE
C       LEVEL:       —
C       LANGUAGE:    FORTRAN
C-------------------------------------------------------------------------------------------
C
C       FUNCTIONAL DESCRIPTION:
C       ROUTINE TO CALCULATE A 16 BIT CYCLICAL REDUNDANCY (POLYNOMIAL)
C       CHECK VALUE BASED ON THE CRC16 POLYNOMIAL,
C       (X**16 + X**15 + X**2 + 1), (WHERE ** INDICATES EXPONENTIATION)
C       FOR A MESSAGE OR A PORTION OF A MESSAGE OF ANY NUMBER OF BYTES
C       IN LENGTH. THIS ROUTINE IS COMPATABLE WITH GRS DATATRAIN II.
C-------------------------------------------------------------------------------------------
C
C       ROUTINE INPUT:
C       NCRC    =  CONTAINS THE PARTIAL CRC RESULT, IF ANY, OF PREVIOUS
C                  CRC CALCULATIONS ON EARLIER PORTIONS OF THE PRESENT
C                  MESSAGE. IF THERE HAVE BEEN NO PREVIOUS CALCULATIONS,
C                  I.E. THIS IS THE BEGINNING OF THE MESSAGE OR A COMPLETE
C                  MESSAGE, THE VALUE MUST BE EQUAL TO ZERO.
C
C       BYTED   =  THE DATA BYTE OF THE MESSAGE TO BE USED IN THE CRC16
C                  CALCULATION.
C
C       ROUTINE OUTPUT:
C       NCRC    =  CONTAINS THE PARTIAL CRC RESULT OF THE CRC16
C                  CALCULATION ON THE CURRENT BYTE PLUS ANY PREVIOUS BYTES
```

```
      C                     IF AN INITIAL CRC VALUE WAS PROVIDED.
      C
      C                     FOR TRANSMISSION:  THE LEAST SIGNIFICANT BYTE OF THE
      C                                        CRC VALUE IS TRANSMITTED FIRST
      C
      C                     FOR RECEPTION:     THE CRC VALUE MUST EQUAL ZERO.
      C
      C        PROGRAM VARIABLES:
      C        BCAR   =   REPRESENTS A CARRY OUT OF FIRST BYTE.
      C        BPAR   =   REPRESENTS THE PARITY OF THE BYTE.
      C--------------------------------------------------------------------------
      C
      C                      LEFT BYTE        RIGHT BYTE
      C        NOTE:  N =   B(2) * ( 2**8 )  +  B(1)
      C
      C        NOTE:  MUST MAKE SURE THAT THE OPERATIONS ARE PERFORMED ON THE
      C               RIGHT BYTE ONLY.
      C
      C        NOTE:  ESTIMATED TIME FOR ONE BYTE IS < 1.1 MILLISECONDS,
      C               ASSUMING 200 NS/CYCLE AND 5500 CYCLES/BYTE.
      C--------------------------------------------------------------------------
      C
      C
               SUBROUTINE   CRC16 ( BYTED, NCRC )
      C
      C--------------------------------------------------------------------------
      $        INCLUDE (:F3:CPYWRT.CLV)
      CRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
      C        REVISION:    00                                  DATE:    JAN. 12, 1984
      CRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
      $        INCLUDE (:F3:IMPLCT.CLV)
      $        NOLIST
      C
               DIMENSION      BT1(2), BT2(2)
               EQUIVALENCE ( BT1(1), NTEMP1 ), ( BT2(1), NTEMP2 )
      C--------------------------------------------------------------------------
      C
               BYTE = BYTED
               NTEMP1 = NCRC
      C
      C        EOR THE LOW BYTE OF CRC WITH DATA = BYTE
      C
               BYTE = (.NOT. BYTE .AND. BT1(1)) .OR. ( BYTE .AND..NOT. BT1(1))
      C
      C        GET CARRY VALUE AFTER DOUBLING PREVIOUS RESULT.
      C
               BT2(2) = 0
               BT2(1) = BYTE
               NTEMP2 = NTEMP2 + NTEMP2
               BCAR = BT2(2)
               BDOUB = BT2(1)
      C--------------------------------------------------------------------------
      C
      C        NOW DETERMINE PARITY ON THE DOUBLED VALUE = BT2(1)
      C        AFTER SHIFTING, TEST LEFT BYTE ONLY.
      C
               BPAR = 0
               D0 100 I = 1,8
               BT2(2) = 0
               NTEMP2 = NTEMP2 + NTEMP2
               IF (BT2(2) .EQ. 0) GO TO 100
               IF (BPAR .EQ. 0) THEN
      C                                                  PARITY = ODD

                      BPAR = 1
               ELSE
```

```
C                                              PARITY = EVEN
                   BPAR = 0
            END IF
  100       CONTINUE
C-------------------------------------------------------------------------
C
C           EOR THE ( LOW BYTE OF CRC WITH DATA ) WITH THE
C                 (( LOW BYTE OF CRC WITH DATA ) * 2 ) = BYTE
C
            BYTE = (.NOT. BYTE .AND. BDOUB) .OR. ( BYTE .AND..NOT. BDOUB)
C-------------------------------------------------------------------------
C
C           THIS TEST SETS BTMP = 0,1,2, OR 3 BASED ON PARITY & CARRY VALUES.
C           P,C = 00        BTMP = 0
C           P,C = 01        BTMP = 2
C           P,C = 10        BTMP = 3
C           P,C = 11        BTMP = 1
C
            IF ( BPAR .EQ. 0 ) THEN
               IF ( BCAR .EQ. 0 ) THEN
                  BTMP = 0
               ELSE
                  BTMP = 2
               END IF
            ELSE
               IF ( BCAR .EQ. 0 ) THEN
                  BTMP = 3
               ELSE
                  BTMP = 1
               END IF
            END IF
            BT2(2) = BTMP
            BT2(1) = BYTE
            NTEMP2 = NTEMP2 * 64
C
C           SHIFT RESULT 6 PLACES LEFT = * 64. HIGH BYTE OF
C           SHIFTED RESULT, BT2(2) = NEW HIGH BYTE CRC VALUE.
C
C           DIVIDE BTMP BY 2.
C
            IF ( BTMP .GE. 2 ) THEN
               BTMP = 1
            ELSE
               BTMP = 0
            END IF
C-------------------------------------------------------------------------
C
C           OR THE LOW SHIFTED VALUE WITH BTMP ( NOTE, BTMP NOW = 0 OR 1)
C
            BYTE = ( BTMP .OR. BT2(1))
C-------------------------------------------------------------------------
C
C           EOR HIGH CRC BYTE WITH PREVIOUS RESULT = NEW LOW BYTE CRC VALUE
C
            BT1(1) = (.NOT. BYTE .AND. BT1(2)).OR.( BYTE .AND..NOT. BT1(2))
C-------------------------------------------------------------------------
C
C           NEW HIGH BYTE CRC VALUE = HIGH BYTE OF NTEMP2 = BT2(2)
C           NEW LOW  BYTE CRC VALUE = BYTE
C
            BT1(2) = BT2(2)
C
            NCRC = NTEMP1
            RETURN
            END
```

Code generated from Fortran

```
FORTRAN-86 COMPILER              GENERATED CODE
:F0:CRC16.F86

                                                  STATEMENT # 1
0002  1E             PUSH   DS
0003  2E8E1E0000     MOV    DS,CS:@@DATA$FRAME
0008  55             PUSH   BP
0009  8BEC           MOV    BP,SP
                                                ; STATEMENT # 10
000B  C45E0C         LES    BX,[BP].@PARAM+0CH
000E  268A07         MOV    AL,ES:[BX].BYTED
0011  88061C00       MOV    BYTE,AL
                                                ; STATEMENT # 11
0015  C45E08         LES    BX,[BP].@PARAM+8H
0018  268B1F         MOV    BX,ES:[BX].NCRC
001B  891E0200       MOV    NTEMP1,BX
                                                ; STATEMENT # 12
001F  50             PUSH   AX                  ; 1
0020  F6D0           NOT    AL
0022  8A1E0200       MOV    BL,BT1
0026  22C3           AND    AL,BL
0028  F6D3           NOT    BL
002A  5A             POP    DX                  ; 1
002B  22DA           AND    BL,DL
002D  0AC3           OR     AL,BL
002F  88061C00       MOV    BYTE,AL
                                                ; STATEMENT # 13
0033  C606050000     MOV    BT2+1H,0H
                                                ; STATEMENT # 14
0038  88060400       MOV    BT2,AL
                                                ; STATEMENT # 15
003C  D1260400       SAL    NTEMP2,1
                                                ; STATEMENT # 16
0040  8A060500       MOV    AL,BT2+1H
0044  88061900       MOV    BCAR, AL
                                                ; STATEMENT # 17
0048  8A060400       MOV    AL,BT2
004C  88061800       MOV    BDOUB,AL
                                                ; STATEMENT # 18
0050  C606180000     MOV    BPAR,0H
                                                ; STATEMENT # 19
0055  C70600000100   MOV    I,1H
           @@000001:
005B  813E00000800   CMP    I,8H
0061  7E03           JLE    $+5H
0063  E93500         JMP    @@000002
                                                ; STATEMENT # 20
0066  C606050000     MOV    BT2+1H,0H
                                                ; STATEMENT # 21
006B  D1260400       SAL    NTEMP2,1
                                                ; STATEMENT # 22
006F  8A060500       MOV    AL,BT2+1H
0073  98             CBW
0074  09C0           OR     AX,AX
0076  7503           JNZ    $+5H
0078  E91900         JMP    ?100
                                                ; STATEMENT # 23
0078  8A061B00       MOV    AL,BPAR
007F  98             CBW
0080  09C0           OR     AX,AX
0082  7403           JZ     $+5H
0084  E90800         JMP    @@000003
                                                ; STATEMENT # 24
```

```
0087   C6061B0001      MOV     BPAR,1H
                                          ; STATEMENT # 25
008C   E90500          JMP     @@000004
             @@000003:
                                          ; STATEMENT # 26
008F   C6061B0000      MOV     BPAR,0H
             @@000004:
             ?100:
                                          ; STATEMENT # 28
0094   FF060000        INC     I
0098   E9C0FF          JMP     @@000001
             @@000002:
                                          ; STATEMENT # 29
009B   8A061C00        MOV     AL,BYTE
009F   F6D0            NOT     AL
00A1   8A1E1800        MOV     BL,BDOUB
00A5   22C3            AND     AL,BL
00A7   F6D3            NOT     BL
00A9   221E1C00        AND     BL,BYTE
00AD   0AD8            OR      BL,AL
00AF   881E1C00        MOV     BYTE,BL
                                          ; STATEMENT # 30
00B3   8A061B00        MOV     AL,BPAR
00B7   98              CBW
00B8   09C0            OR      AX,AX
00BA   7403            JZ      $+5H
00BC   E91C00          JMP     @@000005
                                          ; STATEMENT # 31
00BF   8A061900        MOV     AL,BCAR
00C3   98              CBW
00C4   09C0            OR      AX,AX
00C6   7403            JZ      $+5H
00C8   E90800          JMP     @@000006
                                          ; STATEMENT # 32
00CB   C6061A0000      MOV     BTMP,0H
                                          ; STATEMENT # 33
00D0   E90500          JMP     @@000007
             @@000006:
                                          ; STATEMENT # 34
00D3   C6061A0002      MOV     BTMP,2H
             @@000007:
                                          ; STATEMENT # 36
00D8   E91900          JMP     @@000008
             @@000005:
                                          ; STATEMENT # 37
00DB   8A061900        MOV     AL,BCAR
00DF   98              CBW
00E0   09C0            OR      AX,AX
00E2   7403            JZ      $+5H
00E4   E90800          JMP     @@000009
                                          ; STATEMENT # 38
00E7   C6061A0003      MOV     BTMP,3H
                                          ; STATEMENT # 39
00EC   E90500          JMP     @@000010
             @@000009:
                                          ; STATEMENT # 40
00EF   C6061A0001      MOV     BTMP,1H
             @@000010:
             @@000008:
                                          ; STATEMENT # 43
00F4   8A061A00        MOV     AL,BTMP
00F8   98              CBW
00F9   88060500        MOV     BT2+1H,AL
                                          ; STATEMENT # 44
00FD   8A1E1C00        MOV     BL,BYTE
```

```
0101  881E0400      MOV    BT2,BL
                                              ; STATEMENT # 45
0105  50            PUSH   AX                 ; 1
0106  B84000        MOV    AX,40H
0109  F72E0400      IMUL   NTEMP2
010D  89060400      MOV    NTEMP2,AX
                                              ; STATEMENT # 46
0111  58            POP    AX                 ; 1
0112  81F80200      CMP    AX,2H
0116  7D03          JGE    $+5H
0118  E90800        JMP    @@000011
                                              ; STATEMENT # 47
0118  C6061A0001    MOV    BTMP,1H
                                              ; STATEMENT # 48
0120  E90500        JMP    @@000012
            @@000011:
                                              ; STATEMENT # 49
0123  C6061A0000    MOV    BTMP,0H
            @@000012:
                                              ; STATEMENT # 51
0128  8A060400      MOV    AL,BT2
012C  0A061A00      OR     AL,BTMP
0130  88061C00      MOV    BYTE,AL
                                              ; STATEMENT # 52
0134  50            PUSH   AX                 ; 1
0135  F6D0          NOT    AL
0137  8A1E0300      MOV    BL,BT1+1H
013B  22C3          AND    AL,BL
013D  F6D3          NOT    BL
013F  5A            POP    DX                 ; 1
0140  22DA          AND    BL,DL
0142  0AD8          OR     BL,AL
0144  881E0200      MOV    BT1,BL
                                              ; STATEMENT # 53
0148  8A060500      MOV    AL,BT2+1H
014C  88060300      MOV    BT1+1H,AL
                                              ; STATEMENT # 54
0150  8B060200      MOV    AX,NTEMP1
0154  C45E08        LES    BX,[BP].@PARAM+8H
0157  268907        MOV    ES:[BX].NCRC,AX
                                              ; STATEMENT # 55
015A  E90000        JMP    @@000013
            @@000013:
                                              ; STATEMENT # 56
015D  8BE5          MOV    SP,BP
015F  5D            POP    BP
0160  1F            POP    DS
0161  CA0800        RET    8H


FORTRAN-86 COMPILER               ERROR MESSAGES AND SUMMARY
:F0:CRC16.F86

STORAGE REQUIREMENTS FOR MODULE CRC16:

     CODE AREA SIZE          00164H       356D
     CONSTANT AREA SIZE      00000H         0D
     VARIABLE AREA SIZE      0001DH        29D
     MAXIMUM STACK SIZE      00006H         6D

0 ERRORS DETECTED.
0 WARNINGS ISSUED.
ENTRY POINT IS 2H
COMPILATION OF CRC16 COMPLETE.
```

```
0 TOTAL ERRORS DETECTED.
0 TOTAL WARNINGS ISSUED.

DICTIONARY SUMMARY:
     38KB MEMORY AVAILABLE
      3KB MEMORY USED   (7%)
      0KB DISK SPACE USED

END OF FORTRAN-86 COMPILATION.
```

ANSI-standard Fortran code

```
C ------------------------------------------------------------------------
C   MBYTE   =   BYTED
C   KCRC    =   NCRC
C   NOTE: MAKE SURE THAT THE OPERATIONS ARE PERFORMED ON RIGHT BYTE ONLY
        IMPLICIT     INTEGER*2 (B)
        IMPLICIT     REAL*8 (X,V), REAL*4 (A,C-D), INTEGER*4 (E-F,K-L)
        IMPLICIT     INTEGER*2 (I-J,M-N)
        IMPLICIT     LOGICAL*1 (O-P), CHARACTER*1 (R,T)
        DIMENSION      BTEMP1(2), BTEMP2(2), BTEMP3(2), BTEMP4(2)
        EQUIVALENCE ( BTEMP1(1), KTEMP1 ), ( BTEMP2(1), KTEMP2 )
        EQUIVALENCE ( BTEMP3(1), KTEMP3 ), ( BTEMP4(1), KTEMP4 )
        BYTE    = MBYTE
        KTEMP1 = KCRC
C   EOR THE LOW BYTE OF CRC WITH DATA = BYTE
        KTEMP3 = 0
        KTEMP4 = 0
        BTEMP3(2) = BYTE
        BTEMP4(2) = BTEMP1(2)
        KTMP = 255
        KTEMP3 = IAND ( KTEMP3, KTMP)
        KTEMP4 = IAND ( KTEMP4, KTMP)
        KTEMP4 = IEOR ( KTEMP3, KTEMP4)
        BYTE = BTEMP4(2)
C   GET CARRY VALUE AFTER DOUBLING PREVIOUS RESULT.
        KTEMP2 = 0
        BTEMP2(2) = BYTE
        KTEMP2 = KTEMP2 + KTEMP2
        KTEMP3 = ISHFT ( KTEMP2, 8)
        KTMP = 255
        KTEMP2 = IAND( KTEMP2, KTMP)
        BCAR = BTEMP3(1)
C   DETERMINE PARITY ON DOUBLED VALUE, AFTER SHIFTING, GET RIGHT BYTE
        KTEMP4 = KTEMP2
        BPAR = 0
        DO 100 I = 1,8
        KTEMP4 = ISHFT ( KTEMP4, 1)
        KTMP = 256
        KTEMP3 = IAND ( KTEMP4, KTMP)
        KTMP = 255
        KTEMP4 = IAND ( KTEMP4, KTMP)
        IF (BTEMP3(2) .EQ. 0) GO TO 100
        IF (BPAR .EQ. 0) THEN
C                          PARITY = 1 FOR ODD, = 0 FOR EVEN
            BPAR = 1
        ELSE BPAR = 0
        END IF
100     CONTINUE
C   EOR THE ( LOW BYTE OF CRC WITH DATA ) WITH THE
C   ((  LOW BYTE OF CRC WITH DATA ) * 2 ) = BYTE
        KTEMP3 = 0
        KTEMP4 = 0
```

```
      BTEMP3(2) = BYTE
      BTEMP4(2) = BTEMP2(2)
      KTEMP4 = IEOR ( KTEMP3, KTEMP4)
      BYTE = BTEMP4(2)
C  THIS TEST SETS BTMP = 0,1,2, OR 3 BASED ON PARITY & CARRY VALUES.
C  P,C = 00   BTMP = 0      P,C = 01   BTMP = 2
C  P,C = 10   BTMP = 3      P,C = 11   BTMP = 1
      IF ( BPAR .EQ. 0 ) THEN
         IF   ( BCAR .EQ. 0 ) THEN
              BTMP = 0
         ELSE BTMP = 2
         END IF
      ELSE
         IF   ( BCAR .EQ. 0 ) THEN
              BTMP = 3
         ELSE BTMP = 1
         END IF
      END IF
      KTEMP3 = 0
      BTEMP3(2) = BTMP
      KTEMP3 = ISHFT ( KTEMP3,8)
      KTEMP2 = 0
      BTEMP2(2) = BYTE +  BTEMP3(2)
C  DIVIDE BTMP BY 2
C  HIGH BYTE OF SHIFTED RESULT = NEW HIGH BYTE CRC VALUE.
      IF   ( BTMP .GE. 2 ) THEN
           BTMP = 1
      ELSE BTMP = 0
      END IF
      KTEMP2 = ISHFT ( KTEMP2, 6 )
C  OR THE LOW SHIFTED VALUE WITH BTMP ( NOTE, BTMP NOW = 0 OR 1)
      KTEMP3 = 0
      KTEMP4 = 0
      BTEMP3(2) = BTEMP2(2)
      KTMP = 255
      KTEMP3 = IAND ( KTEMP3, KTMP)
      BTEMP4(2) = BTMP
      KTEMP4 = IOR ( KTEMP3, KTEMP4)
      BYTE = BTEMP4(2)
C  OR HIGH CRC BYTE WITH PREVIOUS RESULT = NEW LOW BYTE CRC VALUE
      KTEMP3 = 0
      KTEMP4 = 0
      KTEMP3 = ISHFT ( KTEMP1, - 8)
      KTMP = 255
      KTEMP3 = IAND ( KTEMP3, KTMP)
      BTEMP4(2) = BYTE
      KTEMP4 = IEOR ( KTEMP3, KTEMP4)
C  NEW HIGH BYTE CRC VALUE = THIRD BYTE OF KTEMP2
C  NEW LOW  BYTE CRC VALUE = FOURTH BYTE OF KTEMP4
      KTMP    = 255
      KTEMP4 = IAND ( KTEMP4, KTMP )
      KTEMP3 = NOT ( KTMP )
      KTEMP2 = IAND ( KTEMP2, KTEMP3 )
      BTEMP2(1) = 0
      KTEMP1 = KTEMP2 + KTEMP4
      KCRC    = KTEMP1
      RETURN
      END
SOURCE STATEMENTS = 97
PROGRAM SIZE = 1176 BYTES
```

With this software package, a PC, and a modest grasp of Basic, users can perform and plot integrals over experimental data.

Mathematical Software in Basic

Dint: Data Integration

David K. Kahaner, Jeffrey Horlick, and Webb L. Wyman
National Bureau of Standards

Many calculations produce a sequence of x, y pairs

$$(x_i, y_i) \quad i=1, \ldots, n \tag{1}$$

where $x_1 < x_2 < \ldots < x_n$. The computational process that generates the data can be complicated, but we assume that the numbers represented in Equation 1 are accurate values of some underlying (unknown) function $f(x)$; that is, we assume

$$f(x_i) = y_i \quad i=1, \ldots, n \tag{2}$$

Dint is a package containing a central computer program to calculate an integral, subroutines to print and plot output, a tutorial, and detailed documentation. The programs are written in Basic to compute an approximation R to the integral I, such that

$$R \approx I = \int_{x_1}^{x_n} f(x)\,dx \tag{3}$$

Problems of the type described in Equations 1-3 are common in engineering and modeling applications. Related to them is the problem of evaluating an integral of a function given by a formula. In a previous article,[1] we described an algorithm and computer program, GLAQ,

for this problem. Both Dint and GLAQ have been designed in the belief that high-quality mathematical software should be reliable, friendly, and portable. (See *IEEE Micro*, October 1983[1] for a discussion of these terms.)

Dint is interactive and automatic. No programming is required. It is only necessary to know enough about Basic to be able to load and execute the program. Dint queries the user about the data and asks questions about calculations and displays. The tutorial included with the package can be retrieved to the CRT screen on command or can be printed out as a text file for examination. (Dint works with or without an attached printer.) Options allow the data points to be entered from the keyboard or a stored file, in sorted or unsorted order. A consistency check is made to be sure there are no repeated abscissas (x values). After the data have been entered, it is possible to examine the data and make corrections. The integral estimate R is accompanied by an error estimate E, such that

$$E \approx |R-I| \tag{4}$$

The values of R, E, and other information are displayed on the CRT screen. A smooth curve through the data can be viewed on systems which support graphics.

Dint was designed for a specific class of problems. Its use is not appropriate if either the x values or the y values are inaccurate. An example is data resulting from experiments. Since Dint creates a function which passes through every x, y pair, experimental error can lead to considerable inaccuracy in the integral. However, Dint is perfectly appropriate for accurate experimental data. The integral over inaccurate data is best dealt with by finding an approximate fit (perhaps by least squares) and integrating that. The plotting feature can still be of use for looking at the shape of the function through experimental data.

Dint's interactive mode is just right for a few data sets, but is awkward for dealing with many sets or for those calculations requiring an integral as an intermediate step. In these cases a user will want to run Dint as a subroutine; and, in fact, the Dint package is actually a collection of separate modules. The central program module performs the integration. Since it contains no printing or plotting operations and has only five input and output variables, it is straightforward to make the module into a callable subroutine. Very explicit directions are given in the detailed documentation which accompanies the package. These directions include exactly which line numbers to delete, which variables to define, and the line number for the GOSUB. All the internal variables are listed. A model calling program is also included. Experience indicates that only modest familiarity with Basic is necessary to utilize this feature, but also that most users will prefer to run Dint interactively.

Dint currently runs on the Tektronix 405X, IBM PC/XT (on the low-resolution text display and the high-resolution graphics display), and the Apple II. Portability is accomplished by using separate machine-dependent subroutines along with the machine-independent central program. Since Dint is written in a small subset of Basic, we expect that only minor changes will be needed to make the package compatible with other microcomputers. These changes are mainly in the syntax of branching (IF) instructions, and in the graphics display commands. Dint, like its companion GLAQ, is in the public domain.*

This article first describes some of the mathematical background used for Dint and then gives a few examples.

Mathematical formulation

An approximation R of Equation 3 is usually designed so that

$$R = \int_{x_1}^{x_n} g(x)\,dx \tag{5}$$

where $g(x)$ approximates $f(x)$. In the case of a method which assumes that the data are accurate, the function g interpolates to f, that is

$$g(x_i) = y_i = f(x_i) \quad i = 1, \ldots, n \tag{6}$$

The functional form of g varies depending on the particular approximation being used. A specific example which ought to be familiar to most readers is that of polynomial interpolation. If $n = 10$ there is a unique 9th degree polynomial

$$P_9(x) = a_0x^9 + a_1x^8 + \ldots + a_9$$

such that

$$P_9(x_i) = y_i \quad i = 1, \ldots, n$$

The a_i's can be computed. They depend on the x's and y's. Thus we can estimate

$$I \approx R = \int_{x_1}^{x_{10}} P_9(x)\,dx = \frac{a_0[x_{10}^{10} - x_1^{10}]}{10} + \ldots + a_9[x_{10} - x_1]$$

High-degree polynomial interpolation is not recommended,[2] but serves as illustration for the more general situation which we now wish to describe.

If we plot the points and connect successive ones by a line segment (dot to dot) we can estimate I by adding the

*Requests for copies should be addressed to D. K. Kahaner at the address given at the end of the article, and must include the model name of the computer, the operating system level, and an appropriately formatted disk or cassette.

area under each of the $n-1$ trapezoids, such that

$$I \approx R = T_n \equiv \frac{1}{2}[y_1(x_2-x_1)+y_2(x_3-x_1)+ \dots + y_{n-1}(x_n-x_{n-2})+y_n(x_n-x_{n-1})] \quad (7)$$

Another popular approximation is Simpson's rule. This is the integral of the function g which interpolates through each successive triple of points with a quadratic.

It should be noted that Equation 7 is a special case of Equation 5. The ensemble of segments connecting y_1, $y_2, \dots, y_n$ is a perfectly good $g(x)$ although the equation for g is unconventional in that there is no single, simple functional form. Rather, between each pair of data points, $g(x)$ is given by a different expression,

$$g(x) = y_i\frac{(x-x_{i+1})}{(x_i-x_{i+1})}+y_{i+1}\frac{(x-x_i)}{(x_{i+1}-x_i)} \quad (8)$$
$$x_i \le x \le x_{i+1} \quad i=1,2,\dots,n-1$$

The trapezoidal rule of Equation 7 has several attractive features:

(1) Simplicity of derivation and use.
(2) Monotonicity and convexity of g on monotonic/convex data.
(3) Accuracy in some special cases.

Item 2 is useful because it guarantees that g follows the data in a very intuitive way, albeit with corners. Smoother interpolants add character of their own: a mixed blessing. By Item 3 we mean that if the x_i's are equally spaced (as is common), and if the underlying function $f(x)$ is smooth and periodic (i.e., $y_1 = y_n$), then Equation 7 is the "most accurate" approximation possible.[3] In those cases where the data is given at unequal spacing, or does not come from a smooth, periodic function, then Equation 7 does not give particularly accurate estimates.

Dint uses as an interpolant

$$g(x) = \sum_{i=1}^{n} y_i h_i(x) + d_i m_i(x) \quad (9)$$

where y_i are the data values, d_i are numbers determined by the code, and $h_i(x)$ and $m_i(x)$ are predetermined "piecewise cubic Hermite" basis functions.[4] This is a common choice and we sketch it now for completeness.

The functions h_i and m_i are defined on the entire interval $[x_1, x_n]$ but each is identically zero on all but a small subinterval. Both h_i and m_i (except h_1, m_1, h_n and m_n) are zero except on $[x_{i-1}, x_{i+1}]$. Between every pair of adjacent abscissas the basis functions are cubic polynomials defined in such a way that

$$\begin{aligned} h_i(x_j) &= \delta_{i,j} \\ h_i'(x_j) &= 0 \\ m_i(x_j) &= 0 \\ m_i'(x_j) &= \delta_{i,j} \end{aligned} \quad (10)$$

The essential feature of this basis is that

$$\begin{aligned} g(x_i) &= y_i \\ g'(x_i) &= d_i \end{aligned} \quad (11)$$

Any selection of the d_i's yields a function g that interpolates the data (x_i, y_i). For example, setting $d_i = 0$ produces a function that is flat through each data point. Regardless of the choice for d_i, the function $g(x)$ has one continuous derivative on the entire interval $[x_1, x_n]$. This is enough to produce a function that is visually smooth.

Cubic spline interpolations can yield spurious oscillations, and can result in an interpolant which is not monotonic even with monotonic data.

With so much choice in the d_i's, we need some additional criteria in order to select the one to use. The plotting feature of the package gives us a clue. Not all combinations of d_i's result in curves that look physically reasonable. For example, there is exactly one choice for the interior d's that gives a function with two continuous derivatives. This is a "cubic spline." In this case d_1 and d_n play a minor role. They simply determine the slope of $g(x)$ at the two endpoints. While a spline is smooth mathematically it may not result in a better looking interpolant. In fact, it is known[5] that cubic spline interpolation can yield spurious oscillations, and in particular can result in an interpolant which is not monotonic even with monotonic data. We think that most users would be unsatisfied to see such an interpolant even if its integral were accurate. However, from the given data (Equation 1) it is possible to compute a set of d_i's that almost always yields an interpolant whose graph is "visually pleasing" or "shape preserving," as discussed in Fritsch and Carlson.[5] Actually there is a family of such sets. Dint uses two different g's from this family, both in the form of Equation 9. Thus at the same time the user obtains both an accurate integral estimate and an acceptable looking plot through the data.

Integrating any g in the form of Equation 9 can be shown to give

$$R = T_n + \frac{1}{12}\sum_{i=1}^{n} d_i\beta_i \quad (12)$$

where

$$\beta_i = \begin{cases} (x_2-x_1)^2 & \text{if } i=1 \\ (x_{i+1}-x_{i-1})(x_{i+1}-2x_i+x_{i-1}) & \text{if } i=2,\dots,n-1 \\ (x_n-x_{n-1})^2 & \text{if } i=n \end{cases} \quad (13)$$

The T_n in Equation 12 is the trapezoidal rule, the right-hand side of Equation 7. Thus the sum in Equation 12 may be thought of as correcting the trapezoidal rule.

From Equation 13 we see that if the abscissas are equally spaced, all the β_i's are zero except for β_1 and β_n. In that case Equation 12 reduces to the trapezoidal rule with endpoint corrections, independent of any choice of the intermediate d_i's. This corroborates the intuitive feeling of many scientists that a rather bad interpolant (say, one using $d_i = 0$) can still give a good integral estimate.

It may be shown that the error in Equation 12 is given by

$$|R-I| = \left| \frac{1}{720} \sum_{i=1}^{n-1} \Delta_i^5 f^{(4)}(\xi_i) + \frac{1}{12} \sum_{i=1}^{n} (d_i - f'(x_i))\beta_i \right| \quad \textbf{(14)}$$

where $\Delta_i = x_{i+1} - x_i$ and $x_i < \xi_i < x_{i+1}$. The derivatives are those of the assumed underlying function $f(x)$, shown in Equation 2. In practice f is not available, but Equation 14 is still useful for understanding the relevant issues. The error depends upon the mesh spacing Δ_i, the smoothness of the function f, and how accurately d_i approximates $f'(x_i)$. Of these only the last is under our control. The most favorable case, never obtainable, would seem to be exact agreement. Actually, one sees from Equation 14 that it is sufficient that the second sum be about the same size as the first; it need not be smaller. Dint computes d_i by a four- or five-point approximation wherever it can or by fewer points if this is not possible. In the code this is referred to as "Newton's difference approximation." This produces the most accurate estimates but may fail to satisfy Item 2 above. Consequently Dint modifies these estimates when necessary.[6] This is still very accurate and gives a good looking interpolant on all but the most pathologic data.

Any program that evaluates integrals needs a reliable process to estimate errors.

In order to provide even more reliability, Dint also includes a second estimate, referred to as the "PCHIP approximation" (Piecewise Cubic Hermite Interpolation Package[5]). Both Newton and PCHIP are of the form Equations 9-11 and usually give nice looking interpolants. Newton is more accurate but PCHIP is guaranteed to be monotonic on monotonic data and hence is a more conservative choice. Either is more accurate than the trapezoidal rule for unequally spaced data. In Dint the user is given an opportunity to choose either approximation method, Newton or PCHIP, and the authors encourage this practice. For most practical problems there is no visible difference between the curves produced, nor do the integral estimates differ significantly.

For equally spaced data, the error is bounded by

$$|R-I| \le \frac{(x_n - x_1)\Delta^4}{720} \max_{x_1<x<x_n} |f^{(4)}(x)| + \frac{\Delta^2}{12}(e_1 + e_n) \quad \textbf{(15)}$$

where e_1 and e_n are the approximation errors of the first and last derivatives, respectively. Thus these endpoint errors can contaminate the integral estimate. The trapezoidal rule of Equation 7 has no endpoint corrections; it effectively takes $d_1 = d_n = 0$. Thus, except for the special circumstance when $f(x_1) = f(x_n)$, the error in Equation 7 tends to be proportional to the square of the mesh spacing. Both PCHIP and Newton give essentially equivalent integral estimates for this equally spaced case, with $|R-I| = 0(\Delta^4)$. Of course, if the data show substantial variation, jumps, etc., the curves produced by these two methods will be different.

One important aspect of any program that evaluates integrals is the need for a reliable process to estimate errors. Equations 14 and 15 describe the error in R but neither of these is computable without knowledge of derivatives of the underlying $f(x)$. Even if f were known, these expressions only provide bounds on the actual error, which can be much smaller than f. When R is computed by either of the two formulas described above, Dint estimates the error by comparison with Simpson's rule, S, such that

$$E = |R-S| \quad \textbf{(16)}$$

For smooth functions R is more accurate than S and so, usually,

$$E \le |R-I| \quad \textbf{(17)}$$

Examples

The following five examples illustrate the output of Dint. For the first four, we used a function whose exact integral could be independently computed and we took, as data, function values randomly selected on the integration interval. The first four computations were run on a Tektronix 4054. This microcomputer has high resolution graphics and hardcopy capability. We chose to use the same type of plotting as for GLAQ,[1] except that the data points are marked on the screen. Simple plots are sufficient for most scientific applications and the "move" and "draw" program instructions are at a low enough level to be available on a wide variety of computers.

Table 1 lists the actual and estimated errors from Dint using both methods of estimating derivatives, PCHIP and Newton. Also included are copies of the interpolant plots (from the Tektronix 4054) to exemplify the plotting and to illustrate that in most cases both approximations produce acceptable interpolants.

Table 1.
Typical Dint Error Estimates.

Example	Function	Points	PCHIP actual	PCHIP estimate	Newton actual	Newton estimate
1	$\int_{-1}^{1} \frac{1}{x^4+x^2+0.9} dx = 1.58223296\ldots$	10	$4\cdot10^{-4}$	$8\cdot10^{-4}$	$4\cdot10^{-4}$	$1\cdot10^{-3}$
2	$\int_{0}^{1} \exp(-x)\,dx = 0.632120558\ldots$	100	$8\cdot10^{-8}$	$8\cdot10^{-8}$	$1\cdot10^{-11}$	$7\cdot10^{-10}$
3	$\int_{-1}^{1} \left(\frac{23}{50}(e^x+e^{-x}) - \cos x\right) dx = .4794282266\ldots$	25	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$4\cdot10^{-6}$	$3\cdot10^{-6}$
4	$\int_{0}^{1} x \sin 10\pi x\, dx = -0.0318309\ldots$	45	$7\cdot10^{-4}$	$1\cdot10^{-3}$	$1\cdot10^{-4}$	$2\cdot10^{-3}$

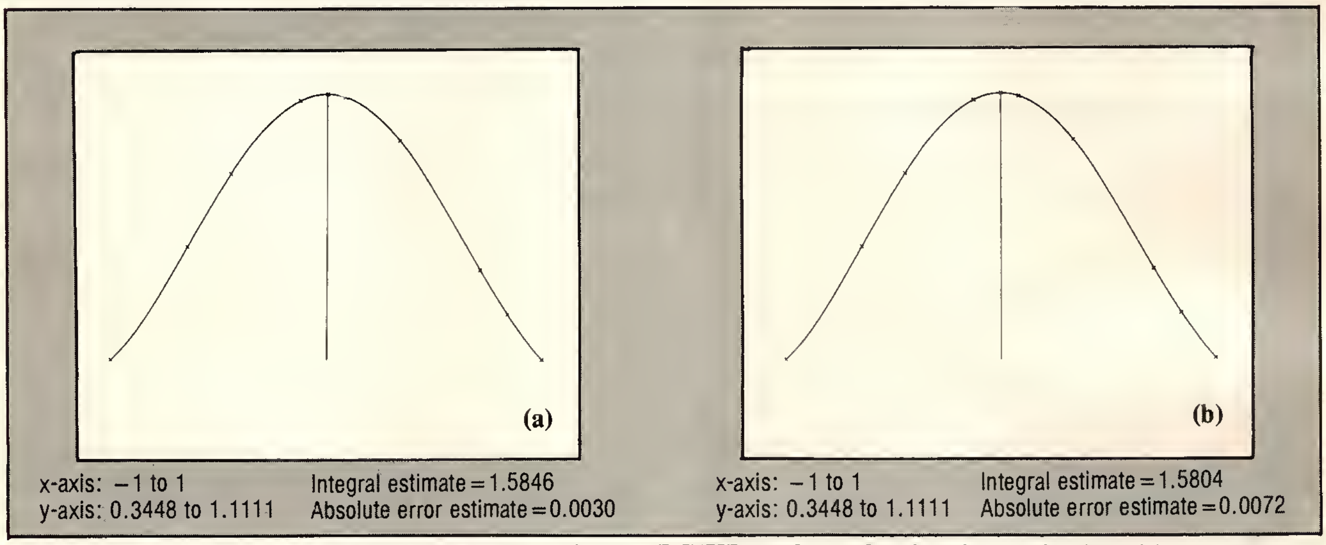


Figure 1. Hardcopy output plotting Example 1 with the PCHIP method of estimating derivatives (a). Example 1, Newton method (b). Figures 1-4 were generated on a Tektronix 4054.

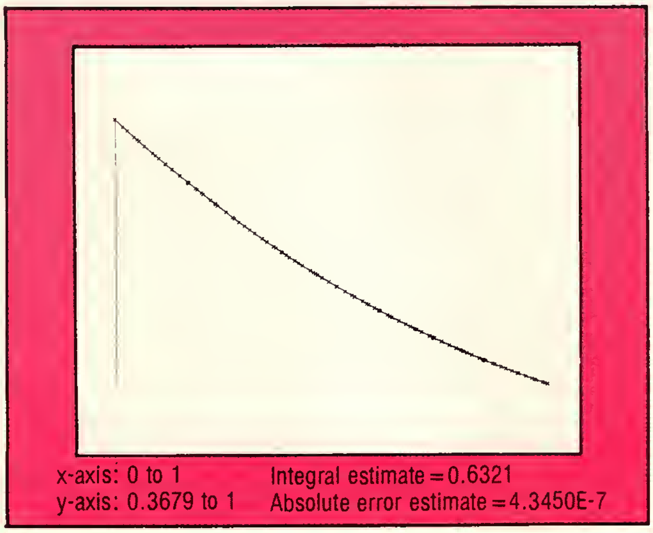


Figure 2. Example 2, Newton method.

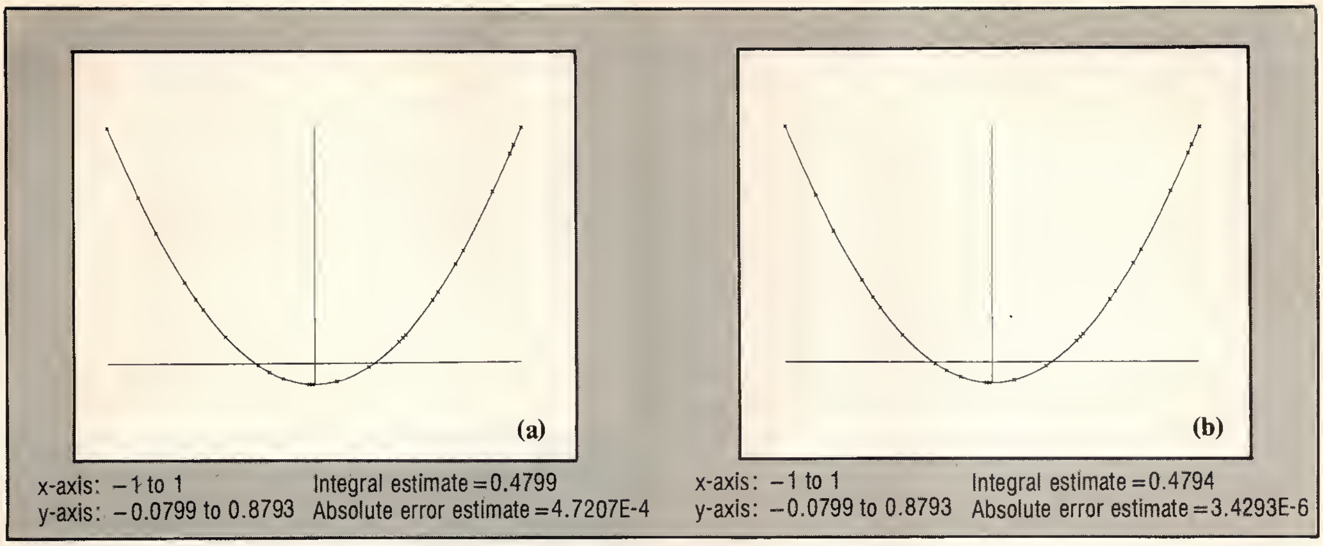


Figure 3. Example 3. PCHIP method (a). Newton method (b).

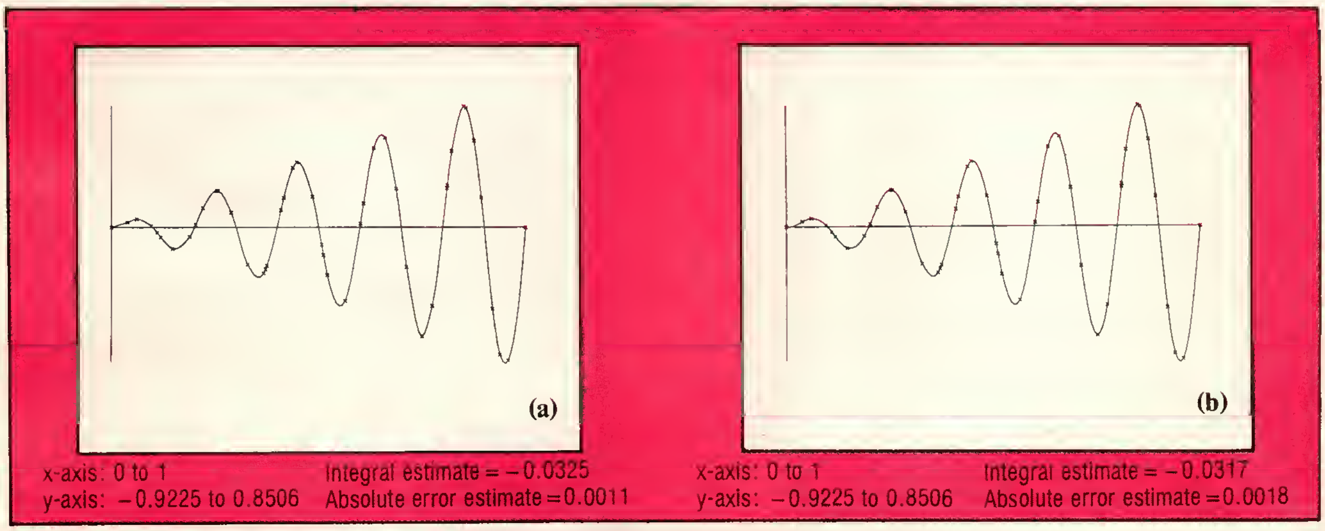


Figure 4. Example 4. PCHIP method (a). Newton method (b).

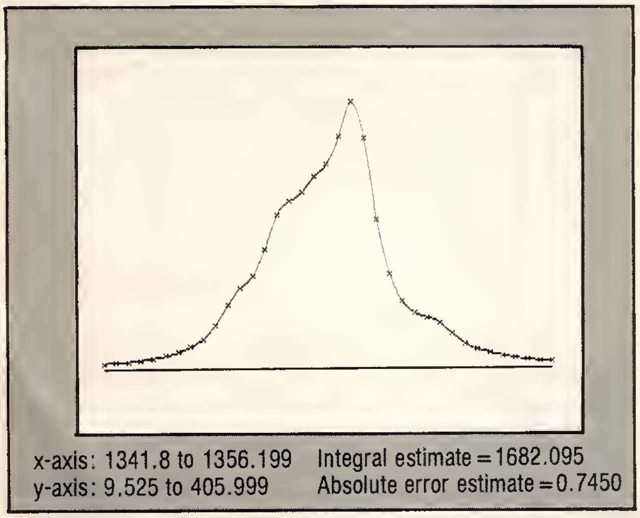


Figure 5. Hardcopy output from an IBM PC/XT plotting holmium oxide photoabsorption values using the Newton method.

Readers are cautioned against inferring too much from these results. Other point distributions give somewhat different estimates. Nevertheless, the general conclusions are that (a) for data from smooth functions the Newton estimates are the more accurate, and (b) the error estimates are reasonably reliable in either case. Finally, we think that astute users will always want to examine the graph of the interpolant. The fifth example illustrates the use of Dint on an IBM PC with high-resolution graphics. The data are values of photoabsorption in a holmium oxide thin film in the region of 5 Å which were generated in the Center for Radiation Research at the National Bureau of Standards. The data are accurate enough to justify the use of Dint. In this case the scientist was particularly interested in the smooth curve Dint generated.

Dint is an example of mathematical software in the Basic language. We used our experiences with an earlier package, GLAQ, to develop it. For example, the user-readable tutorial was only available as a separate text file in GLAQ. In Dint it may also be brought up by direct command from within the executing program. Similarly, textual material on the CRT screen can easily be sent to an on-line printer. This is a common configuration. Nevertheless, we expect that many users will wish to tailor Dint more closely to their particular machine, for example using function keys, split screen for plot and text, etc. With this in mind, even more modularity was designed into the package than into GLAQ.

Acknowledgment

This work is an outgrowth of research in the Center for Applied Mathematics at the National Bureau of Standards. We wish to recognize and thank E. Battiste (C. Abaci Inc.), F. Fritsch (Lawrence Livermore National Laboratories), and S. Haber (NBS) for many useful and significant discussions.

References

1. D. K. Kahaner and W. L. Wyman, "Mathematical Software in Basic—Evaluation of Definite Integrals," *IEEE Micro,* Vol. 3, No. 5, Oct. 1983, pp. 42-46.

2. G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations,* Prentice Hall, New York, 1977, p. 69.

3. P. Rabinowitz and P. Davis, *Methods of Numerical Integration,* Academic Press, New York, 1975, p. 42.

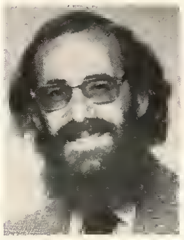
4. M. Schultz, *Spline Analysis,* Prentice Hall, New York, 1973, p. 24.

5. F. N. Fritsch and R. E. Carlson, "Piecewise Cubic Hermite Interpolation Package," SIAM 30th Anniversary Meeting, Stanford University, July 1982. Preprints available as Lawrence Livermore National Laboratory Report UCRL-87285, Livermore, Calif., 1982.

6. J. M. Hyman, "Accurate Monotonicity-Preserving Cubic Interpolation," Los Alamos National Laboratory Report LA 8796-MS, Los Alamos, N.M., 1982.

David K. Kahaner is the technical group leader for scientific software in the Scientific Computing Division, Center for Applied Mathematics at the National Bureau of Standards. His research interests include evaluation of intergrals, solution of ordinary differential equations, interpolation, Fourier transforms, and mathematical software in general. Kahaner received his PhD in applied mathematics from the Stevens Institute of Technology, Hoboken, N.J., in 1968. He spent most of the next 11 years at Los Alamos National Laboratory as a numerical analyst. He has been a visiting professor in the Mathematics Department of the University of Michigan, at the ETH-Zurich, at the Technical University-Vienna, and at the Catholic University of America.

Kahaner's address is Division 713, Room A151, Building 225, National Bureau of Standards, Gaithersburg, MD 20899.



Jeffrey Horlick received his BS in physics from the University of Maryland. His graduate research was conducted at the University of Manchester in England. Horlick has been employed by NBS since 1967, working in the areas of metallurgy, paper physics, and consumer products. For ten years, he worked with the NBS Collaborative Reference Programs, which provide sample materials and statistical analyses for testing laboratory self-evaluation. For the last five years, he has been a Project Leader in the National Voluntary Laboratory Accreditation Program with responsibility for proficiency testing and data analysis.

Webb L. Wyman was graduated with honors from the American University, Washington D.C., where he studied computer science with an emphasis on algorithmic development and applications programming. As a co-op student he worked in the National Bureau of Standards' Scientific Computing Division as a programmer and assisted in the development of numerical quadrature programs. Wyman received an AA degree from Montgomery College in 1980 and was elected to Phi Theta Kappa in 1979.

microNEWS

OA field trials boon to Canadian high tech

by Richard Landry, Managing Editor

Although it has apparently fallen upon hard times with the advent of a new, conservative administration in Ottawa, an office automation research program funded by the Canadian government has proven a boon to that country's fledgling computer industry and has provided government departments with the chance to receive systems configured to their special needs. Whether it also proves worth the investment in time and money made in the selected firms and their products is a question still waiting for an answer, however.

Operating to the tune of about $12 million between the years 1980-1985, the Office Communications Systems Program, funded through the Canada's Department of Communications, has provided a live testing ground for several of that country's office automation firms to develop products that could both satisfy domestic demand and compete in markets abroad. As OCS Director Andre Dubois remarked to a group of American journalists invited recently to inspect the program, that need was felt pressing in an economic environment where Canada was projecting a $20 billion trade deficit with the US by 1990 in this area alone. Moreover, Canadian government, bilingual by law, found American products, particularly software, difficult to adapt to its needs; and in a human interface designed by means of interchangeable tables of commands and prompts rather than by commands embedded in code, it saw the prospect for also fulfilling software needs elsewhere, especially in markets in the Middle East.

For the government agencies involved, however, the more immediate purpose of the trials was to put in place systems, based primarily upon domestic hardware and software, that would provide measurable efficiencies and that could ultimately be generalized to their entire spheres of operation. These agencies—the Department of National Defence, Revenue Canada, Environment Canada, Energy, Mines, and Resources, and the Department of Communications itself—faced the similar task of working with vendors to integrate products that were essentially undergoing a live beta-test into wholly-paper job environments. Likewise, each chose a similar solution, dictated by available technology—Unix-based word processing/electronic mail/database management systems driven by a combination of micros, personal computers, and dumb terminals, and linked by means of PBXs and local area networks.

The most successful systems in the Canadian government's office automation field trials demonstrated the capability for voice/data integration. Shown here: The OCRA/ Gandalf Colleague system on a VT220 terminal with phone add-on.

Most of the problems that arose in the field trials resulted therefore not from implementation of a new technology but from aging in-place technology and from immature system versions. The Department of Communications found that its Ottawa headquarters could not bear the extra electrical burden placed upon it by a

proliferation of PCs on all floors, and a rewiring solution ended up costing ten times more than originally projected. Defence found that excessive heat caused by ventilation system defects brought down its Spectrix 30 micros, despite specially-designed cabinets and fans. Further, the XIOS Renaissance system used by Defence suffered from slow execution caused by problems with implementation of its Xenix kernel, as a XIOS technical manager stated.

Nevertheless, some of the solutions offered by Canadian firms to the departments demonstrated both technical sophistication and maturity. The XIOS system was linked by the Hubnet fiber-optic LAN with an Ethernet back-up, providing the potential for data security offered by fiber optics along with security against system failure offered by the twisted-pair wire. The OCRA/Gandalf system entitled Colleague, designed for Environment Canada, demonstrated smooth implementation of standard Unix facilities along with voice/data integration. Aimed particularly at the US market, the Gandalf system was demonstrated on American-made hardware rather than the domestic variety.

Gandalf's telecommunications-based approach mirrors the strategy of Canadian firms that have successfully penetrated the American market, first through PBX offerings and now through office-automation add-ons. Both Northern Telecom's Meridian product and Mitel's SX-2000 series follow this line, and the feeling among the US journalists assembled on the tour was that this approach most closely matched actual office needs. Mitel representatives cited the IBM-Rolm marriage as a prelude to an American break into this stream. (Previous to the Rolm acquisition, IBM sought partnership with Mitel.)

Beyond the individual successes and failures, however, the Office Communications Systems program points out the peculiar dilemma of such intimate government support for industry. Although for the purposes of the field trial the various systems were implemented on a small scale with leased equipment, nevertheless a switch by any department from a less to a more satisfactory system is bound to entail dislocation and delay. Also, some of the companies involved in the trials have government as their only client, so a departmental pull-out in those cases would probably be politically indefensible, as one government official suggested. Therefore, while some departments will doubtless ride a wave of smooth and efficient office integration as a result of the field trials, others may be saddled with systems too clumsy for needs present or future.

IC copyright deadline July 1

Owners of "mask work" rights in ICs that went onto the market between July 1, 1983, and November 7, 1984, must file their applications for registration in the US Copyright Office before July 1, 1985, or forfeit all rights in those IC layouts. This applies to cells and gate arrays as well as entire random logic chips, and to custom, semicustom, and off-the-shelf ICs.

Section 913 of the SCPA provides that "protection is available under this chapter to any mask work that was first commercially exploited on or after July 1, 1983, and before the date of enactment of this chapter [November 8, 1984]" *only* if application papers in proper form are received in the Copyright Office before July 1, 1985. Ordinarily, that means a properly filled out application form, accompanied by four die examplars, a set of acetate overlays of the mask set (possibly minus ion implant masks), and a $20 filing fee. In the case of ICs commercialized after November 7, 1984, the owner has two years to file an application before rights are forfeited.

Fore previous stories on the SCPA, see *IEEE Micro,* February 1985, pp. 71, 74-75. The June MicroLaw column of *IEEE Micro* will contain a story on problems that cell library owners are having with the Copyright Office's unwillingness to accept registration of those relatively small cells that constitute the majority of cells now in use.

Canada, US take second look at R&D tax credit

by Richard Landry, Managing Editor

The Canadian Department of Communications office automation field trial program is not the only initiative that country has taken to boost its high-tech industries, nor by any means is it the largest. Like the United States, Canada offers a scientific research and development tax credit to those companies, primarily in the computer, pharmaceutical, and medical and industrial instrumentation fields, that depend upon technical advances to stay competitive in international markets. And just as the US is reevaluating this policy in the general press for tax reform, Canada's administration, led by Prime Minister Brian Mulroney, seems bent on trimming and sharpening a potentially powerful instrument of economic stimulus.

According to Robert Beith, Director General for Corporate Rulings at Revenue Canada, the Canadian version of the Treasury Department, the scientific R&D tax program was instituted in November 1983 primarily as a means to allow tax shelters for investors in presumably small, capital-poor high-tech companies and to reduce the tax liability on those companies. The firms could issue a debt instrument, whose purchasers could write it off as a 50 percent tax credit. The issuing firm was then obliged to spend the full amount of the capital raised by this means on R&D, at a tax liability of only 50 percent on every dollar. If the firm did not use the money for research purposes, the full amount was subject to taxes. High-tech firms received no deduction from corporate income tax nor any tax credit under this program.

By contrast, the US R&D tax credit law, bundled into the 1981 Economic Recovery Tax Act, attempts to encourage increased R&D spending by permitting an incremental, 25 percent tax credit for any R&D expenditures in excess of the average shown for the previous three years. As noted by John Stanton, counsel for the Coalition for the Advancement of Industrial Technology, a lobbying organization comprising 52 corporations, 15 universities, and nine scientific/industrial organizations, companies interested in benefitting from the credit must demonstrate to the Treasury Department that their R&D expenditures somehow result in a new product or process. Outright duplications of existing technologies are supposed not to qualify under the law.

Both the Canadian and American programs have drawn fire from critics, although for different reasons. Revenue Canada's Beith noted that some tax-shelter partnerships have taken advantage of a "quick flip" scheme, whereby a high-tech firm sells a share to an investor, at $100 for example, and redeems it shortly from the investor for a lower amount, say $55. The high-tech company then realizes

a profit of $55, clear of all constraints on usage; and the investor, having suffered a net loss of $45 but still in possession of $50 worth of tax credit, realizes a net gain of $5. Stanton said that corporations which do routine software development for internal management or accounting purposes have charged those costs against the credit, under the supposition that they have developed a new product or process.

Canada's solution to this problem, for the moment, has been to place a moratorium on its program until a budget has been hammered out by the new government, sometime in May. (The Canadian Office Communications Systems Program, due to sunset at the field-trial stage in 1985, but with the expectation of continuing through an implementation phase, will apparently not receive further funding.) The US tax credit law, set to expire at the end of 1985, is expected to get a new lease on life in the form of SB 58, introduced by Sen. John C. Danforth (R—Mo.), one of the sponsors of the original provision.

The new version of the law attempts to tighten eligibility requirements by demanding that any expenditures considered must result in a product or process that is "technologically new or improved"—a "term of art," Stanton said, that is meant to cut out routine, in-house software development for other than actual R&D purposes. Software developed for sale is thought to automatically meet this requirement, Stanton said, because "if it is developed for sale, then it has to be innovative in order to compete in the marketplace." The term also covers software developed in-house to drive new equipment or to effect a new process, even though neither the software nor the equipment or process, but rather some end product or process, are for sale. The new version of the law also covers start-up companies, through a carry-forward mechanism, and provides for enhanced tax breaks to firms that do extensive research through universities or through joint-venture companies.

One interesting sidelight to the American R&D tax credit story is that the new Treasury flat-tax proposal, which eliminates most tax benefits to corporations, nevertheless calls for a three-year extension of the program. Stanton said that there are two reasons for this apparent exception to Treasury's overall plan: "It is recognized that companies cannot capture all the benefits of their R&D. Competitors find out what is being done, and so the state of knowledge in general is advanced. Therefore, a subsidy is required to give incentives to individual companies to perform R&D." The second reason makes it even clearer why countries like the US and Canada will continue to encourage R&D, despite problems with administrating laws: "There's the question of international competitiveness. Countries like South Korea and others are not only making use of US technology quickly, but they are giving generous subsidies to their companies to do so."

Open Systems Architecture

by Paul Borrill, IEEE-CS Governing Board

The first meeting of the IEEE Open Systems Architecture Committee met in conjunction with Compcon Spring in San Francisco on Tuesday, February 26. The meeting was attended by experts from all areas of IEEE standardization: networks, software engineering, backplane buses, operating systems, etc. After a morning of presentations about the relationships between standards, the group was unanimously agreed that the IEEE should start a formal activity in Open Systems Architecture—the relationships between functions within a computer system. It was also agreed that, although interconnection between computers would be covered by the scope of the new activity, this would most likely consist of a complete adoption of the ISO's work in the communications area, with their 7-layer OSI mode. An interface would be defined between the IEEE's OSA and the ISO's OSI work.

Most of the afternoon was spent discussing the scope and purpose of the activity. The committee unanimously agreed to the following:

Scope

Information technology standards, including, but not limited to: operating systems, data interchange, networks, buses, I/O interfaces, data storage, man-machine interfaces and languages; taking into consideration security, data integrity, error handling and recovery, binding, naming and addressing. The materials produced by the Group are intended for use by:

- Computer standards groups and bodies.
- Specifiers of systems.
- Vendors of systems.
- Educators.
- System modellers.

Purpose

(1) To facilitate clear descriptions of system behavior.
(2) To classify existing or developing standards.
(3) To identify interfaces where standards should be defined.
(4) To identify area's of overlap and omissions.
(5) The above will be performed in concert with other standards bodies.
(6) This effort will specify internationally acceptable terminology and implementation requirements.

Finally, the group decided to make a formal Project Authorization Request (PAR) to the IEEE Standards Board. This request was approved for submission to the IEEE standards board at the Computer Society's Standards Activity Board/Standards Coordinating Committee, which met in San Francisco the following day.

The next meeting is planned for May 5, 1985, at Stanford University, California. A multi-day workshop is being planned for later in the year. Since a great deal of work is expected to be carried out using electronic mail, it was recognized that effective participation in the Working Group will require members to have some means to communicate via electronic mail.

The Working Group is interested in contacting anyone who is interested in participating in the activity, i.e. becoming actively involved in the design of the reference models for any of the subject areas described above.

Everyone interested in attending the Stanford meeting is recommended to write to the Chairman to ask for details of the previous meeting and agenda.

Please contact:
Paul Borrill,
University College London,
Mullard Space Science Laboratory,
Holmbury St-Mary,Dorking,
Surrey, RH5 6NT, England.

Electronic Mail:
Compmail+ @ P. BORRILL
(CMP0006); UCL-CS.ARPA.

microSTANDARDS

by Robert G. Stewart
Stewart Research Enterprises
1658 Belvoir Drive
Los Altos, CA 94022

In search of excellence in high technology companies

A while back I bought and read the book *In Search of Excellence* by Thomas Peters and Robert Waterman Jr. The book is the result of a provocative search for a new technique to model the better managed corporations in this country. The authors say:

> Good news comes from treating people decently and asking them to shine, and producing things that work. Scale efficiencies give way to small units with turned-on people. Precisely planned R&D efforts aimed at big-bang products are replaced by armies of dedicated champions. A numbing focus on cost gives way to an enhancing focus on quality. Hierarchy and three-piece suits give way to first names, shirtsleeves, hoopla, and project-based flexibility. Working according to fat rule books is replaced by everyone contributing.

Based on my own observations in companies in the high technology area, I'd like to make a few additions to the above list of desirable characteristics. I've noted that American industry is excellent for initiating R&D efforts with its best engineers and scientists. Then, after the product is evolved and put into production, those engineers are supplanted by a new group of production and sustaining engineers. The product tends to remain essentially the same through the years as management ekes out maximum profits from the profit center. It is my belief that this situation is less apt to occur in Japanese firms. A good example is the magnetic tape recorder, which was certainly developed in the US. Note however that the typical tape recorder sold by American firms did not rapidly advance after being placed into production for the mass consumer market.

Compare that situation with how Sony and other Japanese firms finely tuned and improved the tape recorder, making it far better and cheaper. It is now almost impossible to find a tape recorder for the consumer market made in the US. The first home video tape recorder was commercially made by Cartrivision in San Jose and I have one. Its problems led to the demise of the company. Mine no longer works. Compare that with the technically sophisticated VCRs from Japan having good tricolor zoom cameras, timer controls, and remote controls supporting single-frame advance or slow motion. Their tape drive motors have finely crafted servo loops capable of maintaining low flutter and wow and good timing stability. The willingness to upgrade and improve products after they are introduced is an attribute which I believe is too rarely seen in American companies.

Inadequate R&D in American chip houses (despite the government's VHSIC program) has resulted in the 256K bit dynamic RAMs now on the market having Japanese names (with the exception of IBM). It is my observation that this situation is due to two factors: (1) profit centers are controlled by engineers with a circuit or black box orientation, and (2) top management does not foster a research attitude. The latter predicament is supercritical because the beliefs of top management can permeate an entire organization. Such beliefs put a firm brake on the innovation of even the best engineers. A major responsibility of the head of a high-tech company must be to actively foster research and innovation within that company, and to say so loudly and clearly.

At one time, about twenty years ago, the bulwark of semiconductor R&D was the Fairchild Research Lab off Foothill Expressway south of the VA Hospital in Palo Alto. Yet the technology developed in Palo Alto didn't get transferred properly to Fairchild's production facility on Ellis Street in Mountain View. The people I've talked to who were then in the lab blame this on Fairchild's management. Individuals with high school degrees had more say about the fab areas in Mountain View than the research people from Palo Alto. Yet the overall reaction here in Santa Clara (Silicon) Valley was that R&D labs were intrinsically bad, and most chip houses here have avoided them like the plague ever since. Sad. I guess we'll have to get used to reading non-American names on our chips.

One surprise I found in reading *In Search of Excellence* was that the authors proposed a model or "framework" for characterizing businesses which they describe as the Seven S's:

- Structure
- Strategy
- Systems
- Shared Values
- Skills
- Style
- Staff

The authors say: "By recognizing that real change in large institutions is a function of at least seven hunks of complexity, we were made appropriately more humble about the difficulty of changing a large institution in any fundamental way."

Why I was surprised was that years earlier I likewise had generated a list of Seven S's, in this case Smiling Stewart's Sensational Seven S's, describing the ideal woman: Sweet, Soft, Slim, Smart, Sensual, Sex Slave....Somehow I just don't understand why it is that most women I meet regard me as a sexist.

Letter to the editor regarding P896

Dear Dr. Stewart,

I have been following the efforts of the P896 bus standard committee in the press for the last year, and in particular in *IEEE Micro*. Since we will be selecting a 32-bit bus for a new product (Picker manufactures medical imaging systems) within the next few months, I would like to become more familiar with the status of P896 standard. How do I get "plugged in" to the activities and developments on this standard? Questions that I need answers to are:

(1) Is somebody developing support circuits in addition to National Semiconductor? Or has anybody done designs using PALs, etc., that they are willing to share?

(2) Have any companies committed to board-level products based on this bus?

(3) Is there a person or organization that is acting as a "clearing house" for activties and developments related to the standard?

I would appreciate any information you could give me. Thank you.

Yours very truly,
James Pexa
Picker International Inc.
Highland Heights, Ohio

The following response was prepared by Paul Borrill, chairman of the P896 working group:

(1) To get plugged in to the P896 Futurebus activity, write or call

Paul L. Borrill
University College London
Mullard Space Science Laboratory
Holmbury St.Mary, Dorking
RH5 6NT Surrey, England
Country Code (44 in US) 030-670-292

London is five hours ahead of Eastern Standard Time.

Or, if connected to the Computer Society's Compmail + network, send a message to P. Borrill (CMP 0094). Copies of Draft 3 are available from

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, California 90702

for a cost of $10 for members, or $20 for nonmembers.

(2) Four major semiconductor companies have shown interest in producing support devices for the Futurebus—National Semiconductor, Signetics, Monolithic Memories, and Fairchild. However, unlike Multibus II, 896 does not need expensive LSI support chips before boards and systems can be implemented. The bus transceivers are now available from National Semiconductor. The specification for these parts calls for a controlled rise and fall time. The propagation delay of actual parts is running about 9 nanoseconds. The transceiver part numbers are DS3896 and DS3897. (The sharp-eyed amongst you will note an interesting relation between National's part number and another number). National is also considering developing somewhat faster versions of the devices with the low-pass filter and rise-time control circuits removed for use in system buses, which are intrinsically insensitive to crosstalk.

(3) Several companies are planning products based on the Futurebus. Many are already into prototypes, but at the time of this writing (March 2nd) the only company having products on the open market is Tektronix Inc. Other companies will follow in 1985. For more information on Tektronix's products contact:

John Theus
Tektronix, Inc.
PO Box 1000, MS 61-265
Wilsonville, Oregon 97070
(503) 685-2564

Preliminary information indicates that Fairchild is considering bus transceiver parts; Signetics, the protocol chips; Monolithic Memories, the bus transceivers, arbitration chips, and cache controller chips. Tektronix implemented the bus protocol in PALs, but management presently considers them proprietary.

(4) The P896 working group acts as a clearinghouse for activities and developments related to the standard. However, the need for a more commercially oriented activity has been recognized and the working group is now discussing the formation of a separate Futurebus manufacturer's group.

P896 completes its last session

The last meeting of the P896.1 working group was held recently in Santa Clara. The primary concern of the meeting was the proper handling of the cache coherency problem. This problem arises in multiprocessor systems in which individual microprocessors have their own caches. The data in main memory can differ from an updated value stored in some cache memory. If another processor accesses the main memory for that data, not knowing that the value has been modified and resides in another processor's cache, then it will receive the wrong value of the data. The solution taken by the working group was to modify the control codes and to convert an RFU (Reserved for Future Use) line into a Cache Master line. The bus command and status lines provide the minimum hardware capability mandatory for a control protocol to deal with the cache coherency problem. The details of the protocol for using this hardware capability is still under consideration. Rather than hold up the basic standard, the working group decided to issue the basic bus standard as 896.1 and seek Sponsor ballot on it from the Technical Committee on Mini/Micro while the cache protocol is handled separately by a new task 896.2. It is anticipated that this activity will take about 6 months to complete. Two proposed cache protocols were presented at the meeting. The definition of the Control Status Register (CSR) space for the 896 bus will be handled by a new subgroup P896.3.

The modification of the bus protocol to handle caches correctly has been implemented in a manner which will not impact firms already using the prior version. Before the command line EC or CM0 had no use specified. This line, together with a former RFU line now known as CA (for CAche) or CM5, constitute the hardware control lines governing the following protocol:

EC = CM0	CA = CM5	Bus Condition
0	0	Null—do nothing.
1	0	Message passing protocol (as before).
0	1	Current bus master has a cache.
1	1	Cache invalidate—any other master having a cache block corresponding to the current bus address should invalidate that block.

An additional status line, ST2, has been provided for cache coherence. The EV line has been eliminated; disabling of parity checking will now be done by system reset on bus initialize. Parity checking must be explicitly enabled by writing to the CSR space on each board, or by broadcast to the CSR space.

The status-encoding protocol was revised to provide better broadcast response by eliminating unnecessary encodings such as end-of-data on the broadcast cycle, and "busy" on the data cycle.

A second tag bit TG1 was provided for parity checking on the original tag bit TG0. However, both TG1 and TG0 will not be specified except for timing. There will be a recommendation that if parity is required on the TG0 line, then TG1 should be used to provide "odd" parity.

microLAW

by Richard H. Stern
Law Offices of Richard H. Stern
2101 L Street NW, Suite 800
Washington, DC 20037

Source code difference no protection against infringement suit

A federal trial court in Philadelphia has entered the first judgment in a computer program copyright infringement case that did not involve bit-for-bit copying. In *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*, the court found that the defendant's Basic version of a computer program infringed the copyright on the plaintiff's computer program written in Event Driven Language (EDL).[1]

Plaintiff Whelan designed and later succeeded to ownership of a "Dentalab" program for management of dental laboratories. After the sales representation agreement between Whelan and defendant Jaslow was dissolved, Jaslow marketed a Basic version of the computer program. Despite the fact that the two programs were written in different and dissimilar languages, the court found copyright infringement for the following stated reasons:

- The defendant studied "the method and manner that the computer receives, assembles, calculates, holds, retrieves, and communicates data," and studied "the manner in which the information flows sequentially from one function to another."
- This information enabled the defendant to "copy the exact manner of operation for use in a computer that responds to commands written in a different source code language."
- "One thing is readily apparent to even the most casual untrained observation of the operation of the two systems—the visual screens that are displayed are almost identical in format and even in the use of abbreviations and terminology."
- Futhermore, "Prospective users and customers at trade shows found no substantial difference between the Dentalab computer software and the [defendant's] system, and considered them to be the same."

Accordingly, the court enjoined the defendant from further distribution of the infringing computer program and awarded the plaintiff the profits that the defendant earned on the infringing sales.

The trial court may know more about the facts of the case than it is telling us.[2] But judging from the opinion alone, this is a classic case of the court missing the point. One of the basic principles of copyright law is that the copyright in a work protects the author against other people copying the "expression" and not the "ideas" of the work. Only when the expressions in the two works are substantially similar is there copyright infringement; there is no infringement when only the ideas are alike. That means, for example, that a copyright in a book about how to make churns or how to treat cattle for disease does not give the author any rights against people who read the book and then make churns in accordance with its teachings, or who practice the methods of treating sick horses described in the book. A copyright is given to the author without any examination of the prior art in the field, without any study of the qualitative merit of the ideas taught in the book, and is not subject to any requirement that the instructions of the book be sufficiently clear to be understood and followed by readers. A patent, on the other hand, issues only after a government agency has reviewed all of those things. As the Supreme Court said in a case establishing the idea/expression principle in US copyright law, "It would be a fraud on the public" to give copyright owners the rights of patent owners without first subjecting their work to review by an agency like the patent office.[3]

All that the court describes in the present case is the plaintiff's analysis of the copyrighted computer program to determine its ideas, and then the expression of the ideas in a different language. Indeed, the court even remarked that "transferring or converting from one computer language to another is not comparable to translating a book written in English to French."[4] The court noted in this regard that Basic and EDL are so dissimilar that a mere translation from one source code to the other is infeasible, and found that no direct translation had been made.

Moreover, the evidence on which the court relied to find infringement is not probative of copyright infringement. That the screens (displays) are identical in format is immaterial unless the screens are copyrighted, but the plaintiff did not sue under any such copyright. It is well known that the same display can be created by any number of different, unrelated computer programs.[5] The copyright is not on the result but on the way one gets to it. And that potential customers (doubtless, without having access to the codes) found no difference between the two computer programs means only that both programs apparently achieved the same results, not that one code was a copy of the other. Even putting aside the hearsay aspects of the potential customers' supposed analysis of substantial similarity, it is simply not the kind of analysis that permits one to determine whether the two computer programs are so similar in expression that the later one infringes the copyright in the earlier.

In short, the court's analysis is quite defective. Yet it may be easier to find fault

with the analysis than to propose a better one. What should "substantial similarity" have entailed in this case? Can it ever be claimed that one has infringed on expression rather than idea when a program is reworked in another, high-level language? Perhaps not. But that fact, if it is one, is a reason for concluding that copyright law is an inadequate means of protecting important aspects of computer software. That is the real significance of the case. To be sure, there is no justification for the court's disregard of fundamental principles of copyright law in a copyright infringement case. (It would seem that the court overreacted to the defendant's apparently inexcusable claims—not discussed above—to ownership of the computer program after the parties' business relationship broke up.) But the real problem is that even if the court had properly followed copyright law principles, it might not have reached an economically sound result from the standpoint of how best to run a software protection system in the United States. The fault is not that of the courts—it is that of the law.

One question that must be addressed explicitly is whether the following items fall under the protection of copyright law (and if they do not, as is probably the case, ought they to be covered by some other law that ought to come into existence): (1) the data flow of a computer program; (2) the selection of input variables; (3) the selection of matters to be reported as outputs; (4) input and output formats. For which aspects of computer software ought the owner to be compensated when they are used by a later programmer? The principle of denying protection to ideas, while an established part of literary and artistic copyright law for quite sound reasons, requires legislative reexamination in the context of computer software.

"Ought" in this context means that a desired level of progress in the development of software (somewhat difficult to define) would be encouraged and brought about by such compensation and the consequent promotion of security of investment in that software development. At the same time, however, this benefit must more than offset any loss to the public of the benefits of open competition as well as any hindrance to others' programming work.

One answer to some of these questions was put forward in the June through December 1983 *IEEE Micro* Microlaw columns. A proposal was advanced for egislation that would provide new protections against copyright infringement of computer programs (or infringement of computer software protection rights, if the system were placed outside the copyright laws). The standard for determining whether there was infringement was based on whether the second program could be shown to be a "downward translation" of a source code or of an ultra-high-level expression (such as an flowchart, program description, or algorithm) that had been registered before the alleged infringement took place. That concept may be incomplete, however, and more consideration needs to be given to what aspects of computer software ought to be protected (i.e., made subject to compensation or other legal relief if used by others). Again, "ought" is used in the sense of the preceding paragraph.

The enactment of the Semiconductor Chip Protection Act and other recent "pro-technology" measures show that Congress is responsive to the needs of high technology, when a convincing case is made for a change in the law. But the first order of business here would be to formulate sound proposals reflecting a consensus among software creators, users, and marketers, and taking into account the interests of the public in software progress. The editors of *IEEE Micro* share with me the hope that its readers will join in this project. Space in this column, or elsewhere in *IEEE Micro,* will be available to those who have positions to state.

References

1. Opinion of January 22, 1985, as yet unreported.

2. The court received testimony of expert witnesses, but did not rely significantly on it in the opinion. The court observed that it is "difficult for one having practically no knowlege of computers and computer software systems to judge which of two experts in the field is the more credible."

3. In *Baker v. Selden,* 101 U.S. 99 (1879), the alleged copyright infringer used the bookkeeping system described in a copyrighted book, without paying the copyright owner. The Supreme Court held that patent-like protection could be acquired only by those complying with the requirements of the patent laws, not by means of copyright.

4. Compare *Synercom Technology, Inc. v. University Computing Co.,* 462 F. Supp. 1003, 1013 n.5 (N.D. Tex. 1978), in which the court assumed the contrary. The *Whelan* court is probably more correct in its view than the *Synercom* court. Of course, it depends on the language. Perhaps Fortran and Cobol are close enough to each other to sustain a translation while Cobol and APL are too far apart for any translation.

5. This point has been noted in several video game copyright cases involving computer-driven audiovisual desplays. See, for example, *Stern Electronics, Inc. v. Kaufman,* 669 F. 2d 852, 855 (2d Cir. 1982). For example, one can generate an octagon by any of the following methods: (1) generate a square, then subtract triangular bits from each corner; (2) generate a Greek or Swiss Cross, then fill in the triangular spaces between each adjacent pair of arms; (3) generate a square, rotate it around its center by 45 degrees, and display the logical product (set of points enveloping the area enclosed by both figures). The computer program for doing each of these things is different, even though the resulting figures are the same.

   Almost identical computer programs, moreover, can cause very dissimilar displays. See *Midway Mfg. Co. v. Strohon,* 564 F. Supp. 771, 749-53 (N.D. Ill. 1983), in which the court found the two computer programs to be almost bit-for-bit duplicates, although the second video game display (a Pac-Man type game) was unlike that of the original Pac-Man game.

Is microcode hardware or software?

A major controversy over the copyrightability of microcode has erupted between NEC and Intel. NEC plans to market its new V20 and V30 microprocessors in the United States. The V20 and V30 chips are aparently similar to the 8086 and 8088, in that their instruction sets are supersets of the 8086/8088 instruction sets, but the V20/30 apparently also use added instructions from other microprocessors such as the Z80. Probably the marketing idea is to provied a microcomputer that can emulate both IBM PCs and Z80 machines—perhaps other machines as well.

In late December 1984 NEC sued Intel in the San Francisco/San Jose federal court for a judgment declaring that the microcode of the V20/30 did not infringe any copyright in the microcode of the 8086/8088. NEC alleged that Intel had asserted that its copyrights in the 8086/8088 microcode would be infringed by NEC's importation of the V20/30 into the United States. NEC said that the assertions have created in NEC a "reasonable apprehension" that Intel will sue NEC and its customers, involving them in "numerous duplicative and protracted lawsuits involving Intel's spurious copyright claims"

and causing them to suffer "irreparable injury." Consequently, NEC demanded a court declaration that Intel's copyrights are invalid or not infringed, and that Intel has no right to threaten NEC or sue it; NEC also sought a court order prohibiting Intel from suing or threatening to sue NEC or distributors, customers, or users of NEC's products. In late February 1985, Intel replied with a countercharge that NEC had illegally copied Intel's microcode, and Intel demanded damages and an order prohibiting the importation or sale of the chips.

The question of whether copyright law protects microcode from competitive replication has never been presented or decided before. It may be that the court in this case will find that the V20/30 microcode is so different from that of the 8086/8088 that there is no infringement. But if the court instead finds them to be "substantially similar," it will have to decide whether it is legally wrong to copy microcode. The economic implications of a decision either way would be significant—not only in the microcomputer field, but for mainframes as well. It is often difficult and expensive to make an emulator without replicating the microcode, and realizing the same timing with a different microcode is at times said to be virtually impossible; yet, if the timing of an emulator is different from that of the emulated computer, the two machines will not necessarily execute the same computer programs.[1]

What is "microcode"? The term is generally understood to refer to the computer coding for the basic repertory of information processing operations used in the computer's information processor. In the case of these microprocessor chips, that means the basic signal or information processing operations of which the chip is capable. Somewhat more precisely, the microcode of a microprocessor chip is the set of instructions ("microinstructions") implementing arithmetic and logic operations in the chip. The question that the court may well have to decide in the NEC-Intel case is whether microcode is a "computer program," or more generally, whether microcode is the kind of "literary work" that the copyright law protects.

The question is not easy. After all, microcode is more esoteric than the kinds of code involved in prior copyright infringement cases. It is not written by most programmers. Instead, it is written by specialists, considered unconventional even by many programmers, who (as one of them has said) need to work themselves into a "mental state" where they become "a microcode writing machine" before they can produce useful code.[2]

An example of an instruction that is implemented by microcode in Intel's 8086 microprocessor (although it could have been implemented instead by random logic hardware) is the "increment register instruction." This is the instruction for replacing an integer stored in a register by the next higher integer—e.g., replacing 1000 with 1001. The process for accomplishing this may be expressed in conventional English language as follows:

(1) Move the content of register to temporary register *b*; set up Arithmetic Logic Unit to perform an increment of temporary register *b*; prepare to end executing current instruction upon next cycle of clock.

(2) Move Arithmetic Logic Unit result back to register; set flags in accordance with Arithmetic Logic Unit result; end current instruction.

Microcode implementations of such verbal instructions can be spoken of in three senses: (a) source microcode, (b) core dump microcode, and (c) ROM microcode. Source microcode is an assembly code expression stating how to carry out the above steps (1) and (2). Core dump microcode is a written-out string of 1s and 0s that are an object-code form of the source code (i.e., its assembly). ROM microcode is the pattern of open- and closed-circuit transistors within the ROM of the microprocessor, where there is a 1:1 correspondence between open and closed circuits in the ROM, on the one hand, and 1s and 0s in the object code, on the other.

The Intel source microcode for the foregoing pair of 8086 instructions is:

(1) M, tmpb, 1, XI, tb,nx, X
(2) Sigma, M, 4, none, RNI,F

To assemble the source code into an object code (core dump,) each alphanumeric expression in the source code must be replaced by a particular sequence of 0's and 1's, which is defined as equivalent to the alphanumeric expression. In the case of the preceding example, Intel translates "M" to and replaces it by 10010; "tmpb" by 01101; "1" by 01; "XI" by 10001; "tb,nx" by 011; "X" by 0; and so on. Thus, the entire microcode expression above is translated into the following core dump microcode:[3]

(1) 100100110101100010110
(2) 101001001010011110001

The correspondence linking the source microcode, core dump microcode, and the ROM microcode of these expressions is illustrated in Table 1.

The main arguments against according copyright protection to ROM microcode are: (1) it is not intended as a medium of intelligible communication between human beings (as programs in Basic, Fortran, Pascal, Ada, Lisp, etc. arguably are); (2) it is intended primarily to function in the interstices of a machine to carry out the machine's functions (moving signals within the chip, regardless of their assigned meaning, rather than directly manipulating information signals in accordance with the assigned meaning of the signal); and (3) microcode is dictated by the electronic function of the product, with the results that (a) there are only a few ways to write the microcode, (b) the text is an inextricable mixture of unprotectable ideas and only potentially protectable expressions, and (c) protecting the expressions in microcode would impermissibly protect the ideas as well.

Table 1.
Microcode correspondences.

[illegible]	Source code	Core dump	ROM
Move the		1	closed circuit
		0	open circuit
content	M,	0	open circuit
		1	closed circuit
of register to		0	open circuit
		0	open circuit
		1	closed circuit
temporary	tmpb,	1	closed circuit
		0	open circuit
register b		1	closed circuit
etc.	etc.	etc.	etc.

The first two of these arguments are merely a restatement of the general arguments against copyrightablility of ROMed object code. Regardless of the merit, if any, of the arguments, they have not prevailed so far. They were rejected, for example, in the Apple operating system litigation in the Philadelphia and San Francisco federal appellate courts. Intel's argument is that microcode in the ROM of a microprocessor is no different from any other computer program, and

the copyrightability of computer programs was settled by those cases. Consequently, to prevail here, NEC will have the uphill task of presuading the court that microcode is in some way quite different from applications and operating system software. But to any nontechnical court that has to decide one of these cases, the difference between microcode and other computer code will seem negligible. It is all black magic, whether done by microcode-writing cyborgs or other slightly less weird entities, and fine distinctions will probably be incomprehensible and therefore ig nored.

Moreover, Intel can advance persuasive arguments why ROM microcode is no different from other object code. For example, every piece of the microcode in core dump form has a 1:1 correspondence to what looks very much like a message in assembly code. Comparing the source microcode expression "M, tmpb, 1, XI, tb,nx, X," and its assembled version, "100100110101100010110," the *M* and first *10010* both mean "Move the signal in the register"; the *tmpb* and next following numerical sequence, *01101*, both mean "to temporary register *b*"; and so on. Thus, ROM microcode and core dump microcode, which have a clear 1:1 correspondence to one another, seem also to have 1:1, translatable correspondence to more-or-less intelligible-appearing assembly code. In short, assembly microcode looks as much like a medium of communication as any other assembly code; and arguably the same legal principles should apply to both. If other object code is protected under copyright law (now the clear majority view in the United States courts), so too should be microcode. As far as copyrightability is concerned, ROM microcode appears to be in the same category as, for example, the ROM object code of a ROMed interpreter program.[4]

The third NEC argument may go beyond what the earlier cases decided. NEC contends in its complaint that the particular lines of microcode in the ROM of the Intel 8086 are "dictated exclusively by the function to be performed by the microprocessors by means of the Intel microcode." Presumably, the function to which NEC refers is not microprocessing, in general, nor adding, substracting, and comparing, in general. (Intel would argue, of course, that only those should be considered the functions of a microprocessor.) Presumbably rather, NEC argues that the 8086 microcode is dictated by the instruction set of the 8086, and that "NEC is free to use the Intel microcode since it is the sole, or one of the few, possible means of expressing the ideas embodied in it." (The "ideas" are not identified specifically.) In the same vein, NEC argues that there was no new authorship or creativity, and hence no copyrightable "work of authorship" by the microcode programmer, and that the idea and expression are so interrelated in the microcode that the microcode programmer's lines of microcode cannot be protected without at the same time impermissibly monopolizing the ideas.[5]

This aspect of the case may well turn on the particular facts. Thus, the 8086 instruction set may or may not dictate the 8086 microcode. The Intel microcode programmer may have had many possible choices of codes to implement the instruction set, so that it was open to him to have expressed the ideas in a variety of ways; he may have chosen only one way, leaving everyone else free to use the remaining possibilities. If that is so, NEC could use the "ideas" of the microcode without also appropriating the "expression," or particular lines of code. Thus, if Intel could show that there exist (or NEC is unable to show that there do not exist) many alternative possible microcode "expressions" that implement the instruction set, then NEC's functionality and idea/expression arguments would be rejected.

On the other hand, if the instruction set could in fact be implemented only one way (or a few ways), according copyright protection to such a way would, in effect, give a monopoly on the instruction set. If that is done, NEC argues, the copyright law is "unconstitutional because Intel's microcode is, if anything, the 'discovery' of an 'inventor' rather than the 'writing' of an 'author' under the United States Constitution." That rhetoric seems to boil down to a statement that the Consitution demands that instruction sets be kept open unless they can, in some way (which they probably cannot), meet the inventiveness requirements of the patent system. That is an interesting proposition, but also an uphill argument.

Other questions that may be asked are:

- What will be the business consequences of a ruling one way or the other?
- What will be the effect on technological progress in the computer and semiconductor industries? Do the progress incentives and investment security brought about by protection weigh more heavily than any hindrance to latecomer manufacturers?
- How do these considerations affect equipment manufacturers and end users?

Courts do not usually consider such issues in depth when adjudicating lawsuits. But Congress can, should, and presumably does consider it when it amends old laws or passes new ones. By the same token, Congress' inquiry is informed and aided by expert and professional input. In any event, it is at least as important to debate what ought to be as to debate what is. Comments and proposals are, therefore, invited.

References

1. Thus, it has been said that Hitachi and other Japanese manufacturers of IBM emulators were effectively prevented from marketing their products in the United States because they were unable to utilze IBM's microcode or invent around it.
2. T. Kidder, *The Soul of a New Machine*, pp. 97-98.
3. A skilled programmer can write microcode either directly in core dump microcode or else first in source code, which he then translates into core dump microcode. The person who wrote the 8086 and 8088 microcode, I am told, wrote it in source code first and then hand-assembled it into core dump microcode, because that is much easier to do. Intel's copyright registration involved in this case is based on the source code version of the microcode.
4. To be sure, a 1:1 correspondence is not an absolute basis for deciding whether the second corresponding item is protected by copyright law if the first one is. The more sophisticated question asks what is the scope of the protection. Thus, blueprints are copyrightable and so are drawings of chip layouts. Nonetheless, the copyright in either is generally thought to prevent others only from copying the blueprint or drawing as such, but not to prevent others from making the machine or other object depicted in the blueprint or drawing. The 1:1 correspondence between parts of a copyrighted blueprint and parts of the machine depicted in the blueprint does not carry the day in a copyright infringement suit against a rival manufacturer of the machine. (That is, it does not in the United States, though it does in the United Kingdom.)

   Nonetheless, in prior computer program copyright cases, the courts have been inclined to stop their inquiries upon finding a 1:1 correspondence between the source and its complilation. For example, in the litigation over GCA's 99/1010 diagnostic system, the court ruled that, because of a 1:1 correlation, the object code was within the protection given to the copyrighted source code: "Because the object code is the encryption of the source code, the two are to be treated as one; therefore, copyright of the source code protects the object code."
5. Just what the "idea" of the microcode could be is unclear. Perhaps, it is to be equated with implementation of an instruction set, so that "implementing the particular instruction set" is the "idea" and the lines of code used to do so are the "expression."

microREVIEW

PFS:Write Version B

by Ware Myers, Contributing Editor

There I was about six months ago, having just written a very big check (for me) for an AT&T PC6300 (see *IEEE Micro*, February 1985, pp.68-70). Having spent my time studying the hardware side of the purchase, I knew very little about word processing packages; and my only previous experience, many years ago, was with a teletypewriter connected via a modem to a central computer. I remembered that then I had used only a few of the scores of instructions available, so it seemed reasonable to get a "simple" word processing package now. A simple package would probably be low cost, and that was all right, too.

In response to these thoughts my salesman recommended *PFS:Write* (then available as Version A). When I began to use the package, I discovered that it was indeed simple. And, at $140, it was also relatively inexpensive—some word processors sell for $400 to $500.

Later, as I learned more about word processors, I realized that I might well have considered a more sophisticated set of requirements. I might have pondered the type of writing I mainly do. I might have wondered whether the package was easy to learn and use. I might have explored the relationship between the word processor and telecommunications. I might have tried to find out something about its performance capabilities.

Then, just about the time I began to understand that Version A lacked some useful attributes—notably, its working space was rather small in relation to my needs—the vendor announced an enhanced package, Version B, and offered to replace the program disk for recent purchasers. I made the exchange, but first let me review my needs as I now understand them.

Type of writing

Articles for the magazines of the IEEE Computer Society are composed mainly of straight text, divided into sections by two or three different type fonts. Footnotes, figure captions, and references fall outside the straight text. In addition, there are tables, photographs, and diagrams. Sometimes equations or mathematical formulas are needed.

Text. *PFS:Write* does straight text just fine. It performs the editing commands summarized in *Package at a Glance.* It supports directly four type styles, both to the monitor and the printer: regular, boldface, underlined, and a combination of boldface underlined. So each section or subsection of an article can be distinguished by a different type style.

Separate text. *PFS:Write*, unlike some word processing packages, does not automatically keep track of footnotes, figure captions, and references. Of course, footnotes can be typed on the screen at the bottom of the manuscript page to which they apply, but this course seems somewhat foolhardy. That bottom shifts up and down as additions and deletions to the manuscript are made or as the page layout is reformatted. One could wait until the draft is final to locate the footnotes, but experience indicates that occasions to revise the text keep occurring, even with a draft believed to be final. Moreover, the eventual location of the footnote in the typeset version is unknown anyway. The best solution probably is to bracket the foonote text so that it stands out from the body of the article.

In the case of captions and references, it is usually best to write them at the same time that the pertinent text is being written. They can be located, in the form of a separate paragraph, at the end of the text paragraph to which they refer. Then, at the time of the so-called final draft, the MOVE procedure can be used to transfer them to the end of the article. *PFS:Write* does not automatically number and renumber figures and references as additions and deletions are made, so this task must be done manually.

Tables. *PFS:Write* provides tabs that cause the cursor to stop on the initial letter and on the decimal point. The width of a table is limited to wherever the left and right margins are set; 10 and 70 are the default values, and the maximum margins are 1 and 78. To create a table wider than the margins being used for the rest of the document, one would have to establish a separate document which, at the time of printing, could be JOINed to the basic text at the desired location within the text. Making a table wider than the screen is not possible; that is, *PFS:Write* does not support horizontal scrolling. To keep the table all in one piece vertically, as the

document is revised and page boundaries change, the NEW PAGE command (to the printer) can be employed.

Diagrams. Of course, as a word processor *PFS: Write* does not draw diagrams. An integrated companion program, *PFS: Graph*, prepares bar charts, pie charts, and plots from data contained in *PFS: File*. For drawings such as block diagrams, some other package would be necessary. For the present I plan to continue making my occasional diagram by hand on paper.

Equations. Fortunately—from a typing standpoint—I employ few equations. With superscripts and subscripts and complex space arrangements, they have always been hard to create on a typewriter. Our typesetter finds them hard to set in type. In my own case, I have often left space in the manuscript and entered them by hand. Such items are also a problem on a word processor. *PFS: Write* does not have direct superscript and subscript commands, but it does have a command that transmits codes for them to the printer. It may very well be necessary to treat complex equations in the same way that I intend to handle diagrams—on separate hand-drawn pages.

Easy to learn and use

Easy to learn. Everyone claims this virtue. Moreover, after we have used a package for several months, we tend to forget how much trouble it originally gave us, and we begin to think it was easy to learn. Is there a more objective way to evaluate this factor? After contemplating my navel, I decided that there is.

Obviously the manual should be well written—grammatical, jargon-free, well organized, indexed, etc. In my judgment, *PFS: Write* meets this standard.

Beyond good writing, the manual should lead you into what you need to know to start working before it presents more advanced topics. In general, the *PFS: Write* manual does this. After a short introductory chapter, the manual first deals with "getting started." Then comes a chapter on simple editing, providing a minimum set of commands that enable you to do real work. In fact, with just this much information my wife, who is allergic to high technology, wrote a 2000-word letter (I had disconnected our electric typewriter, forcing her onto the computer). *PFS: Write* saves the more advanced editing capabilities—search and replace, block editing, emphasizing text, and indenting margins—until Chapter 7.

Of course, as the somewhat sated owner

Package at a Glance

PFS:Write, Version B — $140 — Released October 1984

Software Publishing Corporation
1901 Landings Drive, Mountain View, CA 94043

General

Prerequisites	Personal computer with MS DOS operating system. At least one double-sided disk drive. Printer.
Systems tested by vendor	IBM PC with MS DOS 1.1, 2.0, or 2.1 IBM PC XT with MS DOS 2.0 or 2.1 IBM PC Portable with MS DOS 2.1 IBM PCjr with MS DOS 2.1 (80 column only) Compaq Portable with MS DOS 1.1 or 2.02 Compaq Plus Portable with MS DOS 2.02 Columbia Data Products MPC with MS DOS 1.25 Corona PC with MS DOS 1.25 Eagle PC with MS DOS 1.25 Hyperion with MS DOS 1.25H Panasonic Sr. Partner with MS DOS 2.0
System tested by reviewer	AT&T PC6300 with MS DOS 2.11
Integrated with	PFS:Proof (spelling checker), PFS:File, PSF:Graph, PFS:Report, PFS:Plan, and PFS:Access
Working space	64,000 characters

Editing functions

Cursor moves	By character, word, or line, or to the beginning or end of text or to user-set tab positions, both tabular and decimal.
Wordwrap	Moves automatically by whole word to next line or next page.
Scrolling	Up or down a line or a page at a time.
Delete	A character, a word, an entire line, a labeled field, or the whole text.
Insert	Add characters, leaving the existing material unchanged, or overwrite existing material.
Move	Label block of up to 100 lines and move to new location.
Copy	Label block of up to 100 lines and copy to one or more new locations, while leaving original in place.
Search	Find a specified string, replace the strings one at a time after the user concurs, or replace all instances of the string automatically. It can conduct a so-called "wild card" search, where only part of the string is specified.
Count words	Approximately 15,000 words per minute.
Disk functions	Save working file. Save working file as an ASCII file for telecommunications purposes. Retrieve named file, files produced by other packages, or ASCII files. Append disk file to working copy. Remove file from disk.
Format page	Create one- or two-line headings and footings, and set them left, right, or center. Omit headings and footings on first page. Number pages automatically. Save page definition as a file for future use. Force new page when printing. Set left, right, top, and bottom margins and number of lines of text per page. Format one line left, right, or center. Indent left margin temporarily.
Printer control	Print single or double space. Justify right margin when printing. Print screen. Join additional files to working copy when printing. Print specified page or pages or continuously. Embed printer control codes in text, or specify on print menu for entire document. Shift left margin to right a specified number of spaces, centering document on page or making two-column format possible (by passing paper through the printer twice).

of eight new manuals dealing with various aspects of my equipment and programs, I am finding that, reader though I normally am, I try to avoid these manuals. *PFS: Write* also provides a series of help screens, keyed to the particular menu under consideration; it supplies a cardboard cutout to identify the function keys; and it notifies you of a problem by means of some 53 error messages in reasonably plain English.

For instance, it may say, "The document is too large to Append." The manual elaborates further: "The document you tried to Append will not fit in the working copy. Either resave the document in smaller sections and then Append again, or use the JOIN command to append it when printing."

Easy to use. Again, this concept is fairly subjective; but, at the very least, editing commands should be easy to remember. There are four ways of interacting with *PFS: Write*: menus, named keys, function keys, and printer commands embedded in the text.

The program comes up on a Main Menu that can be reached at any time by pressing the ESC key. It presents a choice of five secondary menus. Menu selections are made by keying in the number of the choice. No memory strain here.

The 15 keys at the right of the IBM-type keyboard used by *PFS: Write* are identified by names or icons. No memory problem here.

The 10 function keys at the left of the keyboard are labeled F1, F2, etc., but the cardboard cutout serves as an *aide memoire*. If you lose it, one of the help screens has the same information. No problem here.

The commands affecting the printer operation are expressed in words, such as JOIN, JUSTIFY, NEW PAGE. and PRINTER, set off from the normal text by asterisks to denote that they are commands. These words, of course, are easy to remember. Unfortunately, the PRINTER command must be followed by a control code that is more difficult to remember. There may be scores of such codes denoting type fonts and sizes, superscripts and subscripts, and other capabilities of the printer. Moreover, the control codes have to be expressed in decimal characters corresponding to the position of the characters in the ASCII table. Even if you could remember the ASCII characters for a few capabilities, you would have difficulty remembering the decimal equivalents. Printer control codes could be a memory problem.

Another aspect of the easy-to-use concept is the degree of complexity involved in implementing procedures that require a series of commands to effect. A procedure such as labeling and moving a block of text to a new location inherently involves several steps. The best a program can do is to emulate the logical progression you would follow naturally as you think your way through what has to be done. The MOVE sequence, for example, proceeds as follows:

- Bring the cursor to the beginning of the text you want to move.
- Label the first letter (with a function key).
- Label the rest of the line and successive lines of the block (with the Return key).
- Transfer the block (up to 100 lines) to a buffer (with the Delete key).
- Move the cursor to the new location.
- Enable the insert function (with the Insert key).
- Transfer the block from the buffer to this location (with the Duplicate function key).

Telecommunications

It seems that each word processor package employs different embedded format instructions, so only like packages can fully reconstitute the incoming message on the screen. However, all word processors are supposed to be able to read ASCII files, which are stripped of these instructions. So, if you wish to telecommunicate, your word processor should be able to create ASCII files. *PFS: Write* can, and the enhanced version does it by the simple technique of having you add the extension .ASC to the existing disk file name, and then storing the new file to disk.

If you wish to telecommunicate, *PFS: Write* can create ASCII files.

New features

The principal enhancement in Version B of *PFS: Write* is the doubling of the working space. Formerly 32,000 characters, it is now 64,000 characters. I used the program's word counting command to ascertain the size of the particular document with which I filled the working space. The count was 10,884 words. The counting time (on the AT&T PC6300) was 44 seconds, or about 15,000 words per minute, compared with about 2000 words per minute on Version A. Since the hardware is the same, the algorithm must have been greatly improved. Another capacity enhancement is the expansion of the block buffer from 35 lines to 100 lines. Also it can be set up to operate with either a monochrome or color monitor.

Enhancements. Version B offers only half a dozen editing improvements, so *PFS: Write* is still a simple word processor. One of the new features is an indent capability that allows the left margin of a portion of the text within a document, such as a quotation, to be moved to the right a specified number of spaces. Another is that, where headings and footings were previously restricted to a center position, Version B adds a choice of left corner or right corner as well. Headings, footings, and page number may be omitted on the first page of a document.

A dubious improvement is the ability to justify the right margin when printing a document. The improvement is dubious because the program merely adds whole spaces between the words on the line in question (*PFS: Write* does not have word division). Sometimes the added spaces become so large as to be irritating. Indeed, the manual recommends that word justification be used sparingly in long documents because "research shows that justified text is harder to read. Varying line lengths help people keep their place when reading."

While *PFS: Write* does not automatically save the working copy to disk at intervals, the new version has made it unnecessary to type the file name each time. The system now remembers the file name when you indicate that you wish to save the working copy.

Another new feature is the ability to select *PFS: Proof*, a proofreading product, directly from a menu.

If you use the same format for many documents, it can be a nuisance to set it up each time on the page definition menu. The enhanced product permits the format to be stored on disk and simply recalled when needed.

Performance. Formerly the help screens were recalled from disk upon request, in an operation that took about four seconds. Now they are kept in RAM and come up instantaneously. Transfers to and from the block buffer are also instantaneous. Page down or page up takes about half a second. Time to transfer to or from disk, however, appears to have nearly doubled: saves to about 16 seconds and retrievals to about 9 seconds.

SMI develops C-Link application generator

Software Manufacturers, Inc., developers of the Basic-to-C Language translator, S-Tran, has announced C-Link, a C application generator.

C-Link is an application generation system that combines the translation capabilities of S-Tran with development aids to produce C application programs for execution under Unix operating systems. Utilizing C-Link enables a programmer to develop programs in the C language by writing either in Basic or directly in C with the help of a library of Basic statements and functions, the company claims.

C-Link accepts programs through keyboard entry or transfer from other systems, producing functionally equivalent programs in C. The resulting C programs are then compiled and linked to form an application system that executes under Unix or Xenix, including the newly released version 3.0.

Price for C-Link is $695.00 in single quantities. Delivery is off-the-shelf.

For further information, contact Software Manufacturers, Inc., 20270 S. Leapwood Ave., Carson CA 90746; (213) 538-8174.

Reader Service Number 14

Coprocessor brings IBM compatibility to Macintosh

Dayna Communications announced a product which enables the Apple Macintosh personal computer to use software written for the IBM PC. The new product, called MacCharlie, permits Macintosh users to access the large library of IBM PC-compatible software. MacCharlie also permits Macintosh users to connect to IBM PC serial networks and to use IBM PC-compatible printers.

MacCharlie is a separate hardware unit which functions as a coprocessing device. The user connects MacCharlie to the Apple Macintosh with a provided cable. MacCharlie causes the Macintosh to act like an IBM PC or a standard Macintosh, depending on the mode the user selects.

Dayna Communications is located at 50 South Main Street, Suite 530, Salt Lake City, Utah 84144; (801) 531-0600.

Reader Service Number 15

Image Technology's ImageAction software, combined with the company's PC-VISION Frame Grabber, permits image digitization and display on the IBM line of personal computers.

Software and frame grabber allow image digitization on PC

A software product that brings interactive image processing capabilities to personal computer users was announced by Imaging Technology Incorporated. In a related announcement the company introduced two hardware enhancements to its PCVISION family.

The new software product, called ImageAction, enables IBM PC, XT, and AT computers to be used by non-programmers for image processing applications in business, industry, medicine and research.

Used in concert with Imaging Technology's PCVISION Frame Grabber—a device that converts a standard analog video camera signal to digital data—ImageAction can be used to perform image processing and graphics functions.

An optical "mouse" control allows users to select from a variety of image processing functions. The high-resolution monitor connected to the PC-VISION Frame Grabber displays original and processed images, as well as the menus used to select image processing functions.

The software works on an "area of interest" principle. Instead of having to alter peripheral or background parts of a given image, ImageAction can focus on a particular subsection of the image.

In addition to the interactive calling of image processing functions, ImageAction has an online Help facility.

ImageAction works with Imaging Technology's PCVISION Frame Grabber, a video digitizing board that plugs into one of the slots inside an IBM PC, XT, or AT computer. The Frame Grabber allows these computers to acquire and store images for processing or manipulation, then display the images on a video monitor with a resolution of 512×480 pixels.

Two PCVISION Frame Grabber enhancements—eight-bit digitization and pseudocolor capabilities—were also announced by Imaging Technology. Compatible with earlier Frame Grabber versions, the enhanced hardware board displays monochrome input as pseudocolors, making greater resolution image analysis possible, the company says. Included with the pseudocolor option are four input lookup tables which can be used to transform the image prior to storing it.

Image Technology Incorporated is located at 600 West Cummings Park, Woburn MA 01801; (617) 938-8444.

Reader Service Number 16

Softwire data transmission system announced

A serial data transmission system that uses a facility's power wiring has been announced by Compu-Mech, Inc. The system, called SOFTWIRE, is intended for industrial data collection and control applications. It is said to eliminate the need to run expensive wiring throughout a facility. SOFTWIRE routes multiple serial ASCII channels through an in-plant power system (up to 480vac). Handshaking, error checking and retries are all done by the SOFTWIRE controllers, making the system transparent to the transmitting and receiving devices.

Applications include driving remote printers, displays, or data collection and control equipment. The company says that any application that requires serial ASCII transmissions at rates up to 4800 baud can use the SOFTWIRE system. Combining SOFTWIRE with Compu-Mech's 2WIRE data collection and control modules creates a system that can be driven by a single serial port on a plant (or personal) computer.

SOFTWIRE utilizes a power-line-carrier design that is intended to overcome typical problems in industrial environments, such as long transmission distance and noisy electrical circuits. In addition, SOFTWIRE supports transmission media other than the power line, such as twisted pair or radio links.

For free literature on the SOFTWIRE data transmission system, write or call Compu-Mech, 5242 Angola Road, Suite 75, Toledo, Ohio 43615; (419) 535-6702.

Reader Service Number 17

S-100 slave card permits three users

Advanced Digital Corp has announced the availability of the Multi-Slave S-100 slave processor board.

Designed to operate with both 8- and 16-bit master CPUs, the Multi-Slave accomodates three users with three Z-80H (8MHz) CPUs, 128K of memory and two serial I/O ports per user. The Multi-Slave operates with TURBO/DOS and NETWORK-OS operating systems.

According to the company, Multi-Slave is designed to take two less S-100 slots than a three-card set. S-100 bus systems can thus grow as large as 30 users for a 12-slot mother board, the company continues.

Multi-Slave is priced at $1450.00 retail and is available from stock. Advanced digital is located at 5432 Production Drive, Huntington Beach, CA 92649; (714) 891-4004.

Reader Service Number 18

Mimic Master brings fault tolerance to local area networking

Digital Microsystems, Inc., has introduced Mimic Master, providing "hot backup," or redundant processing capability, on the company's HiNet Local area network. Mimic Master enables fault tolerance, or continuous network operation in the event of unanticipated hard disk or master computer failures.

The Mimic Master, sometimes referred to as a shadow master, "mimics" or duplicates any data written to the network (primary) master, in realtime. All of the "mimic" capability is performed in software that emulates a fault-tolerant mode.

Mimic Master attaches to the network system in three ways. The user can boot it as Mimic, install it as a replacement for primary master, or implement it in workstation mode. Requirements for Mimic mode operation call for the Mimic to contain an identical copy of data files, partitions, default settings and system tables from the master computer.

A network of up to 63 stand-alone PCs or HiNet intelligent workstations—running any combination of CP/M-80, CP/M-86, MS-DOS, or PC-DOS operating systems—can be controlled and supported by the network or the Mimic Master, the company says. On HiNet, the primary master continuously polls the network, including the Mimic, for its various data processing needs. The Mimic detects the master computer and "listens" to the network's "traffic" at all times during network operation.

In the event of disk or master computer failures, the master station is turned off and the Mimic is reset. When the Mimic detects that the master computer is absent, it acts as the primary computer on the network. This capability enables network users to locate and repair failed system components while the network continues its operation.

Additional hard disk storage and streamer tape systems are available as options for the primary and mimic masters. The Mimic Master is currently available starting at $7950. For more information on Mimic Master or the HiNet LAN system, contact DMS, at 1755 Embarcadero, Oakland, CA, 94606; (415) 532-3686.

Reader Service Number 19

System access manager designed to prevent piracy

An access management and user authentication system, primarily designed to eliminate unauthorized access to computer systems, networks, and programs, has been introduced by Gordian Systems, Inc. Hand-held, domino sized Access Keys which work in conjunction with computer resident protection routines, are the new product's major elements. The use of this product, the company says, will significantly reduce the ease and frequency of computer piracy, theft, and other abuse.

When held to the screen of a computer system that is protected, the company's Access Key reads information which has been generated by the matching protection routines. The proper password is determined by the Access Key using the information that was read from the screen. This password, which is different for each use of the computer, is displayed on the key itself. The password is good only for that access of the computer. When typed into the system and accepted by the protection routines, the correct password permits user access. The product can also be used to protect individual software programs.

Each key contains optical sensors, a custom-integrated circuit, a five-year battery, a six-character alpha-numeric liquid crystal display, an internal clock, and counter components.

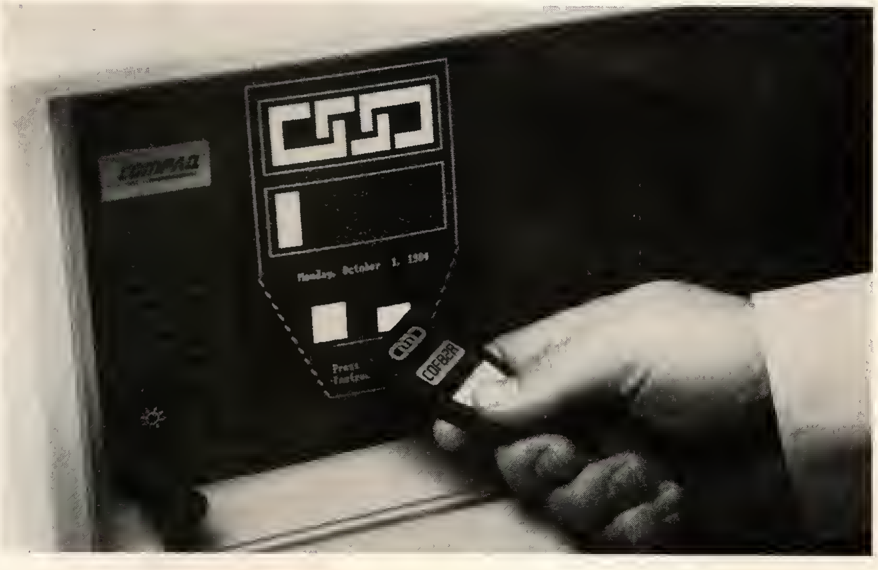
Fixed password systems are currently the primary methods of managing access to computer systems and networks. This approach, claims Gordian Systems, has proven highly vulnerable. The fixed password remain valid for long periods of time, and can often be determined "hackers" who use other computers to generate and try all possible password combinations.

Other conventional approaches, the company says, such as call-back modes, answer-back terminals and hardware integrated and dependent add-on devices, have been viewed as overly expensive, difficult to manage, user inconvenient, and susceptible to forced entry and breakage.

Alternatively, Gordian Systems' approach delivers password uniqueness. The valid password changes with each attempted use of the computer. The only way to determine the valid password for that particular use of the computer or data is by possessing the matching Access Key. The password that the Access Key generates is created in part from a computer-generated random number that cannot be predicted, and in part from Access Key-resident data that is said to be even more difficult to determine. This Access Key data is what individually matches the key to a specific user or system. Since the password is different for each attempted use of the computer, there is no way to try all possible passwords. Passwords cannot be written down, nor can they be distributed on an electronic billboard.

The Gordian Systems Access Key is priced depending upon quantity. The company is located at 3512 West Bayshore Road, Palo Alto, CA 94303; (415) 494-8414.

Reader Service Number 20



The Gordian Systems Access Key is designed to ensure data integrity by generating a unique access code for each use.

Apple announces Appletalk personal network

Apple Computer, Inc. has introduced AppleTalk, a personal network for the Macintosh Office computers and related peripherals. The AppleTalk Personal Network lets personal computers share high-performance peripherals and connects computers within a work area of approximately 1,000 feet for a suggested retail price of $50 per connection. AppleTalk can also serve as a tributary system, using bridges and gateways to link to other networks.

A single AppleTalk network connects up to 32 devices, with the computers and peripherals configured in any combination. Personal-computer users can share peripherals such as file servers and Apple's LaserWriter printer.

Apple also designed AppleTalk to interact with other networks. Through intelligent bridges, users will be able to connect two or more AppleTalk networks to form larger networks. AppleTalk networks will also be able to communicate with other networks through intelligent gateways.

Because the circuits to support AppleTalk were built into the Macintosh family of products at the outset, users do not have to open their machines to add cards to connect to AppleTalk. The AppleTalk Connector Kit consists of a connection box and cables to link a device to the network. AppleTalk is said to need no network administrator to install or configure it.

Apple's software design conserves the network's data bandwidth so that data can be transferred at speeds comparable to those of networks with much higher bandwidths, the company claims. Six Kbytes of software (for the Macintosh) are needed to implement the basic network protocols.

The devices connected to an AppleTalk Personal Network exchange data over a shared, shielded, twisted-pair cable. AppleTalk controls traffic using a software protocol called carrier-sense, multiple access with collision avoidance (CSMA/CA), which was designed to allow all devices to compete equally for access to the cable.

With CSMA/CA, a computer or peripheral device that has data to transmit along the network first "tests the water" by "sensing" whether the line is busy or free. If other traffic is already on the bus, the computer or peripheral device waits until the line is free before transmitting.

To avoid collision, the devices connected to AppleTalk follow a collision-avoidance routine. Upon sensing that the line is free, a network device waits 400 μsec plus an additional time, determined randomly by the software, before reserving the line via a quick "handshake process." This process keeps other devices from using the line at the same time. Other devices seeking access to the cable must continue to test the line until it is free, at which point they can begin the collision-avoidance cycle to reserve the network for their transmissions.

The suggested retail price of $50 for AppleTalk includes the AppleTalk connector and two meters of cable. Additional ten-meter cables and connectors may be purchased. AppleTalk is available begining March 1985 in the United States and Canada through all Apple distribution channels, and in June 1985 internationally. During the second quarter of 1985, 100-meter cable custom-wiring kits will also be available. Apple Computer, Inc., is located at 20525 Mariani Ave., Cupertino, CA 95014; (408) 973-2042.

Reader Service Number 21

Data compressor transmits at up to 9600 bps

RAD Computers, Inc. has announced the CompressoRAD-2, a high-speed asynchronous data compressor which can be used to reduce communication cost by compressing the transmitted data at ratios of 2:1 or 4:1.

The unit compresses asychronous data at speeds up to 19200 bps, and it transmits the compressed data over synchronous modems at speeds up to 9600 bps.

In addition to compression, the CompressoRAD-2 also performs async-to-sync conversion and error control.

The CompressoRAD-2 uses an adaptive compression algorithm that creates special codes, optimized for the data being transmitted. This enables effective compression of any type of data such as text, numerics, and data bases, the company says.

Error-free data is accomplished by the use of an ARQ protocol between the two compressors. End-to-end transparency is maintained and flow control is used, allowing the system to operate at two to four times its original speed without loss of data.

RAD Computers is located at 40 N. Van Brunt St., Englewood, NJ 07631; (201) 568-1466.

Reader Service Number 22

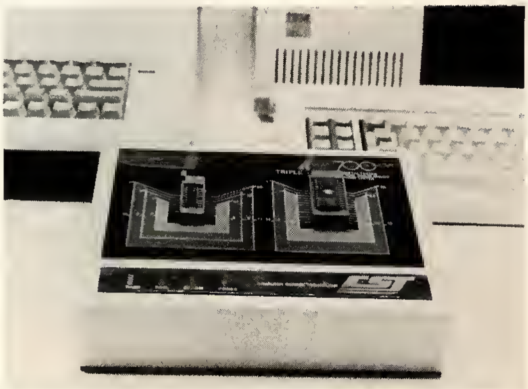
Circuit tester works with Apple II or IBM PC

The Triple Crown 700 was recently introduced by Computer Service Technology as a multifunction IC tester, EPROM programmer, and memory tester using the personal computer (Apple II or IBM) as the host. For IC testing, the unit has a built-in 700 device library and automatically identifies the IC being tested. Software allows the user to program-in his own test table for new ICs and custom ICs. EPROM programmer features include read, write, edit, compose, and compare functions for EPROMS up to 265K. Memory testing accepts a user input table or a table read off another device.

The system comes with system console, software diskette, and user programming manual.

As an introductory offer, the Triple Crown 700-AP to be used with the Apple II is priced at $495 each, and the Triple Crown 700-IBM for the IBM PC at $595 each. Both units are available from Computer Service Technology, 13350 Floyd circle, Dallas, TX 75243.

Reader Service Number 23



The Triple Crown 700 tester is designed for circuit test with the IBM PC or Apple II.

IEEE MICRO'S PRODUCT SUMMARY

Product Summary Editor: Victor P. Nelson

For more information, circle the appropriate RS No. on the Reader Service Card at the back of the magazine.

Boards

Manufacturer	Model	Comments	R.S. No.
Apricorn 7050 Convoy Court San Diego, CA 92111 (619) 569-9483	Super Serial Imager printer interface card	Card offers capability to transfer high-resolution images from Apple II series computer screens to Apple Imagewriter or comparable serial printers. Firmware also supports 300/1200 baud intelligent modems from Hayes, Novation, and other firms, without need to load software communication package onto system.	41
Cubit, Inc. 190 S. Whisman Rd. Mountain View, CA 94041 (415) 962-8237	Model 7500 family STD Bus-based CPU boards	Series of boards is based on the 6502 8-bit microprocessor. Models employ NMOS and CMOS technology, are available with 56 to 72 parallel lines, 0 to 2 serial lines. Model 7550 features color graphics CRT and floppy disk controllers. Start at $230/board.	42
Data-Sud Systems 2219 S. 48th St. Ste. J Tempe, AZ 85282 (602) 438-1492	DSSEDPDX family dual-ported RAM modules	Modules may be accessed either from VMEbus or VMXbus. Single-slot boards offer jumper-selectable VMEbus memory access times of between 150 and 300 ns, between 180 and 330 ns for VMXbus. Model DSSEDPDX-1 supplied with 512 Kbytes 64K x 4 SIP dynamic RAM, DSSEDPDX-2 offers 1 Mbyte capacity. Unit quantity price for DSSEDPDX-1 is $1595. DSSEDPDX-2 is priced at $2695.	43
Extended Systems 6062 Morris Hill Lane PO Box 4937 Boise, Idaho 83711	LaserConnection family printer controllers	Controllers permit the Hewlett-Packard LaserJet printer to be interfaced with IBM systems. ESI-2646 interfaces to Displaywriter system; ESI-1312 permits applications with DisplayWrite 2 software family; ESI-2613, which includes ESI-1312 capability, permits LaserJet to be shared by up to three PCs equipped with parallel printer ports; and ESI-2617, which also includes ESI-1312, permits three PCs with serial ports to use LaserJet. $1695 for singles; discounts on quantities.	44
Heurikon Corp. 3201 Latham Drive Madison, WI 53713 (800) 356-9602	HK68 10Mhz UNIX board	Upgrade to an earlier version running at 8Mhz, board features 6800/6810 CPU, MMU, up to 1Mbyte dual-access RAM with parity, up to 64 Kbytes EPROM, four RS-232-C ports, Winchester and streamer tape drive I/F, and two iSBX connectors. Runs on Multibus, supports UNIX System III and V with drivers for Ethernet and floating point processors.	45
Interphase Corp. 2925 Merrell Road Dallas, TX 75229 (214) 350-9000	Maverick SMD PC-80 disk controller	Permits IBM PC/AT and AT&T PC 6300 to use Storage Module Device (SMD) drives, which are said to offer up to five times the speed of a PC/XT with Winchester disk. Single-board controller is offered alone or as part of company's RDS 350 and 375 and FDS 590 SMD subsystems. Board priced at $1295 singles, $700 OEMs.	46
Wave Mate, Inc. (213) 978-8600	Bullet-286 motherboard	An 80286 retrofit for the 8088-based IBM PC/XT, this board provides up to 640 Kbytes on-board RAM with zero wait states. Upgraded systems are claimed to provide greater compatibility with the standard PC and PC/XT than does the PC/AT. Available in four models, with either 256K, 512K, 640K, or 1 Mbyte RAM, priced at $1995, $2395, $2495, or $2995, respectively.	47

Manufacturer	Model	Comments	R.S. No.

For more information, circle the appropriate RS No. on the Reader Service Card at the back of the magazine.

Chips/Components

Manufacturer	Model	Comments	R.S. No.
Microsync, Inc. PO Box 116302 Carrollton, TX 75011	dClock clock-calendar	Designed for the IBM PC and compatibles, product installs directly onto system motherboard and automatically enters time and date each time computer boots up. Internal battery maintains product function during system shut-down. Retail price $59.95.	48

Peripherals

Manufacturer	Model	Comments	R.S. No.
Data Terminals & Communications 590 Division St. Campbell, CA 95008 (408) 378-1112	Octave intelligent laser printer	Priced at $7995, this system can be used independently by up to eight IBM PCs. Printer controller is based on 10MHz 16-bit processor with 500 Kbytes EPROM and 2 Mbytes virtual bit-map memory, and communicates with PCs via standard RS-232 serial ports. A 128-Kbyte input buffer permits queueing of print jobs. Engine produces eight pages per minute at resolution of 90,000 dots per square inch. Bundled with LIASON software for capabilities such as automatic collation, multiple copy, system control.	49
Interface, Inc. 21101 Osborne St. Canoga Park, CA 91304 (818) 341-7914	Models DMA10 and ATDMA10 removable Winchester disk	Designed for IBM PC and PC/AT, subsystem promotes multiuse/multitasking/data security. Each cartridge holds 10.5 Mbytes memory, fits into B drive of unit. DMA10 for PC priced at $2295, ATDMA10 for PC/AT priced at $1695.	50
Matrix Instruments, Inc. 1 Ramland Road Orangeburg, New York 10962 (914) 365-0190	Videowriter 832 digital-to-analog interface	Color graphics interface is designed for use in networked environment. System accepts commands in standard ASCII format, sends to devices requiring RGB input. In conjunction with microcomputer, system can operate with analog film recorder to obtain up to 832 x 630 resolution output, with projection system for live data presentations, or with modem. Features 16 simultaneous colors from a palette of 4096.	51
Otari Data, Inc. 271 N. Mathilda Sunnyvale, CA 94086 (408) 738-4808	C-200 series half-height 5.25-inch Winchester drives	Series features ST506/412 interface, average access time of 85 milliseconds. Drives conform to physical dimensions of half-height 5.25-inch form factor. Series includes C-214 10-Mbyte and C-226 20-Mbyte drives. C-226 units available at less than $500 in OEM quantities.	52
Remote Measurement Systems, Inc. 2633 Eastlake Ave. East, Ste. 206 Seattle, WA 98102 (206) 328-2255	ADC-1 data acquisition and control system	Analog-to-digital data acquisition and control system features 16 analog input channels of 13-bit resolution (12 bits plus sign), 4 digital inputs, and 6 controlled outputs. Also serves as controller for BSR X-10 series AC line carrier remote control modules. Unit designed with CMOS technology, functions as RS-232 peripheral unit, tested compatible on 20 brands of microcomputers. Can be battery operated for use with portable micro. Price: $395.	53
SWI International Systems 7741 East Gray Road Ste. 2 Scottsdale, AZ 85260-3496 (602) 998-3986	C*Vue flat panel display	Liquid crystal display is designed to mount on Apple IIc, features 80 character x 24 line resolution with graphic display capability. Draws power from IIc via cable or from power pack. Display retails at $599, power pack at $159.	54

Manufacturer	Model	Comments	R.S. No.
For more information, circle the appropriate RS No. on the Reader Service Card at the back of the magazine.			
Software			
Interactive Concepts 1161 N. El Dorado Place, Ste. 340 Tucson, AZ 85715 (602) 296-9977	71 B-Talk interface	Program for the IBM PC or PC/XT permits users to write programs for Hewlett-Packard HP-71B hand-held computer. Allows program upload/download. Runs on PC-Dos 2.0. HP-71B with HP-IL, HP-71 Forth Assembler ROM and HP-IL/RS-232C interface required. $75.	55
Systems			
AST Research 2121 Alton Avenue Irvine, CA 92714 (714) 863-1333	jrCombo multifunction module	Snap-on module enhances IBM PCjr by allowing memory expansion of up to 512 Kbytes RAM, parallel printer/plotter port, and clock-calendar device. Bundled with company's SuperPak/jr software utilites package. Priced at $395, $695, or $1395 for 128, 256, or 512 Kbyte versions, respectively.	56
CAD Counsel 231 East Lemon Monrovia, CA 91016 (818) 359-6091	Protean + computer graphics system	Turnkey drafting and design system dedicated to applications including PCB design, schematics and mechanical design, facilities planning, mapping, and architectural design. IBM PC and PC/XT compatible, system includes 8088 microprocessor, 8087 numerical processor, 512 Kbytes RAM expandable to 960K, 10 Mbyte hard disk drive, serial and parallel printer ports, and Autocad 2 design and drafting software. Price: $7950.	57
Carroll Touch PO Box 1309 Round Rock, TX 78680 (512) 244-3500	Smart-Frame scanning infrared touch input system	New version of company's standard system eliminates controller board requirement. Surrounding CRT or flat-panel display, microprocessor-embedded system frame creates a matrix of infrared beams. When matrix is broken by finger or stylus, system sends command to host computer. Available in 9-, 13-, and 19-inch formats for $450, $595, and $795, respectively.	58
Douglas Electronics, Inc. 718 Marina Blvd. San Leandro, CA 94577 (415) 483-8770	Ap'seed "building block" dedicated systems	System permits use of Apple II series bus convention for dedicated applications. Minimum required components are integrated by means of Ap'seed mother board or ribbon bus. Components include CPU board, 64K RAM board, EPROM board, power supply, bus expander, writable ROM board, among others. Prices available upon request.	59
Rabbit Associates, Ltd 4F, No. 400 Keelung Rd. Sec. 1, Taipei, Taiwan, ROC TLX 13078 GRETANG	R-85 personal digital image processor	This standalone device has the capability of processing digital images for analog transmission via modem. It performs RS-170 video signal digitalization in real time, image display with pseudo color, and digital storage and retrieval. System configured with 8088 CPU and 8087 floating-point processor, 256 Kbytes RAM, two 360-Kbyte floppy disk drives, monochrome alphanumeric and color image display monitors. Runs MS-DOS. Priced at $895 US for single quantities; OEM discount available.	60
Racal-Dana Instruments 4 Goodyear Street Irvine, CA 92718 (714) 859-8999	Model 1150 Data Acquisition System	System provides capability to collect analog and digital data, compare them with previously programmed limits, generate control actions, and report collected data and control actions over IEEE-488 GPI bus. Up to four time periods may be programmed within 24 hours. Scan interval, report interval, active channels, and initial state of output can be independently programmed for each time. Service requests can be generated on out-of-limit data. Price: $1895.	61
Scott Instruments 1111 Willow Springs Drive Denton, TX 76201 (817) 387-9514	VET-232 voice entry system	Speaker-dependent, isolated-word voice-entry terminal provides voice data entry for all systems supporting RS-232C data terminals. Incorporates Motorola 6809 processor and A/D converter with proprietary recognition preprocessor. System sends user-defined vocabulary to host computer as a character string. User may designate up to 57 vocabulary entries of up to 20 characters in length each and up to 1-1/2 seconds duration. Additional memory permits up to 199 vocabulary entries. Priced from $2995 to $4595, depending on quantities.	62

PROFESSIONAL CALENDAR

Conferences sponsored or cosponsored by the IEEE Computer Society are indicated by the society's logo. For listing in Calendar, submit information **four weeks before the month of publication**—e.g., for the **August 1985** issue, send information by **June 1**. Send to *IEEE Micro,* Calendar, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

May 1985

Third Annual Microelectronic Engineering Conference, May 1, Rochester, New York. Contact Lynn Fuller, Microelectronic Engineering Program, Rochester Institute of Technology, 1 Lomb Memorial Dr., PO Box 9887, Rochester, NY 14623; (716) 475-2035.

ACM Symposium on Small Systems, May 2-3 (tutorials on May 1), Danvers, Massachusetts. Contact Fred Maryanski, EECS Dept., U-157, University of Connecticut, Storrs, CT 06268.

End User Computing: The Changing Role of the Systems Professional and the End User (ACM), May 2-3, Minneapolis, Minnesota. Contact Robert P. Bostrom, Indiana University, Business 574, Bloomington, IN 47405; (812) 335-8449.

Comdex Spring, May 6-9, Atlanta, Georgia. Contact The Interface Group, 300 First Avenue, Needham, MA 02194; (617) 449-6600.

Fifth International Conference on Distributed Computing Systems, May 13-17, Denver, Colorado. Contact Earl Swartzlander, TRW (02/2791), 1 Space Park, Redondo Beach, CA 90278; (213) 535-4177.

CICC 85, Custom Integrated Circuits Conference (IEEE), May 20-22, Portland, Oregon. Contact Aris K. Silzars, Tektronix, Inc., PO Box 500 MS 13-800, Beaverton, OR 97077; (503) 627-6980.

ECC 85, 35th Electronics Components Conference (IEEE), May 20-22, Washington, DC. Contact J. A. Woolley, 3M Co., 3M Center, Bldg. 207-1W-10, St. Paul, MN 55144, or Tom Pilcher; (317) 261-1592.

Sixth IFAC Workshop on Distributed Computer Control Systems, May 20-22, Monterey, California. Contact R. E. Bonivert, Lawrence Livermore National Laboratory, PO Box 5508, L-469, Livermore, CA 94550; (415) 422-5416.

Trends and Applications 85, Utilizing Computer Graphics (National Bureau of Standards), May 21-22 (tutorials to be offered May 20), Silver Spring, Maryland. Contact Trends and Applications 85, c/o IEEE Computer Society, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO, or Mark Skall, National Bureau of Standards; (301) 921-2431.

1985 ACM Sigmod, International Conference on Management of Data, May 28-31, Austin, Texas. Contact Sham Navathe, Computer and Information Sciences Dept., 512 Weil Hall, University of Florida, Gainesville, FL 32611; (904) 392-7440, or Gene Lowenthal, MCC, 9430 Research Blvd., Echelon I, Suite 200, Austin, TX 78759; (512) 343-0860.

June 1985

Tutorial Week Spring 85, June 3-7, San Francisco, California. Contact Tutorial Week San Francisco, IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; (714) 821-8380.

ISCAS 85, International Symposium on Circuits and Systems (IEEE et al.), June 5-7, Kyoto, Japan. Contact Takeshi Yanagisawa, Tokyo Institute of Technology; Faculty of Engineering, 2-12-1, Ookayama, Meguroku, Tokyo 152, Japan; phone (03) 726-1111, ext. 2560.

Fourth Canadian CAD/CAM and Robotics Conference, June 18-20, Toronto, Canada. Contact Conference Office, Canadian Institute of Metalworking, 1276 Sandhill Dr., PO Box 7317, Ancaster, Ontario L9G 3N6, Canada; phone (416) 648-5011.

ICC 85, IEEE International Conference on Communications, June 23-26, Chicago, Illinois. Contact John D. Johanneson, Midwest College of Engineering, PO Box 1147, Lombard, IL 60148; (312) 627-6854.

29th International Symposium on Mini- and Microcomputers and Their Applications (ISMM), June 25-28, Sant Feliu de Guixols, Gerona, Spain. Contact E. Luque, Universidad Autonoma de Barcelona, Facultad de Ciencias, Departamento de Electricidad y Electronica, Bellaterra, Barcelona, Spain; phone (3)-6920200, ext. 1356.

July 1985

NCC 85, National Computer Conference (ACM, AFIPS, DPMA, SCS), July 15-18, Chicago, Illinois. Contact AFIPS, 1899 Preston White Dr., Reston, VA 22091; (703) 620-8900.

Siggraph 85 (ACM) 12th Annual Conference on Computer Graphics and Interactive Techniques, July 22-26, San Francisco, California. Contact Ellen Frisbie, Siggraph 85 Conference Services Office, 111 E. Wacker Dr., Suite 600, Chicago, IL 60601; (312) 644-6610.

WCCE 85, World Conference on Computers in Education (IFIP, AFIPS), July 29-August 2, Norfolk, Virginia. Contact Richard L. Dobson, Jr., AFIPS, 1899 Preston White Dr., Reston, VA 22091; (703) 620-8935; TWX 710-833-9037.

August 1985

Fourth ACM Sigact-Sigops Symposium on Principles of Distributed Computing, August 5-7, Minaki, Ontario, Canada. Contact Michael A. Malcolm, Faculty of Mathematics, Dept. of Computer Science, University of Waterloo, Ontario N2L 3G1, Canada; (519) 885-1211.

September 1985

Euromicro, 11th Symposium on Microprocessing and Microprogramming, September 3-6, Brussels, Belgium. Contact Euromicro, Attn. Chiquita Snippe-Marlisa, p/a TH Twente, Dept. INF, Room A306, PO Box

217, 7500 AE Enschede, The Netherlands; phone +(31) (53)−338799; telex 44200 thes.

Compint 85, First International Conference on Computer-Aided Technologies (ACM), September 10-12 (tutorials to be offered September 9), Montreal, Canada. Contact Stephen G. Leahey, PO Box 577, Desjardins Postal Station, Montreal, PQ H5B 1B7, Canada; (514) 870-8842.

Ninth Data Communications Symposium (ACM), September 10-13 (tutorials to be offered September 9), Whistler Mountain, British Columbia, Canada. Contact W. P. Lidinsky, AT&T Bell Laboratories, 6B309, Naperville-Wheaton Rd., Naperville, IL 60566; (312) 979-6246.

International Seminar on Computer Networking and Performance Evaluation (INRIA, IFIP, et al.), September 18-20, Tokyo, Japan. Contact Yutaka Takahashi, Dept. of Applied Mathematics and Physics, Kyoto University, Kyoto 606, Japan.

October 1985

Office Automation 85 (ACM), October 2-4, Erlangen, West Germany. Contact H. Wedekind, IMMD VI, Martensstr. 3, D-8520 Erlangen, West Germany.

10th Conference on Local Computer Networks, October 7-9, Minneapolis, Minnesota. Contact Dan E. Gahlon, Interactive Systems/ 3M, 225-4S-06 3M Center, St. Paul, MN 55144.

Compsac 85, October 7-11, Chicago, Illinois. Contact Compsac 85, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

Computer Graphics Atlanta 85, Conference and Exhibition: Using Computer Graphics to Enhance Productivity and Management Effectiveness (NCGA, WCGA), October 20-24, Atlanta, Georgia. Contact Conference Director, CGA 85, 2033 M St., NW, Suite 333, Washington, DC 20036.

Infomatics 85, October 21-24, Amsterdam, The Netherlands. Contact Program Committee, Infomatics 85, PO Box 34404, Bethesda, MD 20817.

First Pacific Computer Communication Symposium (ACM et al.), October 22-24, Seoul, Republic of Korea. Contact K. H. Kim, University of South Florida, Tampa, FL 33620; (813) 974-4184.

Symposium on Expert Systems in Government, October 23-25, McLean, Virginia. Contact Marshall Abrams, Mitre Corp., 1820 Dolley Madison Blvd., McLean, VA 22102; (703) 883-6938.

November 1985

Ninth Annual Symposium on Computer Applications in Medical Care (ACM), November 10-13, Baltimore, Maryland. Contact Michael J. Ackerman, SCAMC-Office of CME, George Washington University Medical Center, 2300 K. St., NW, Washington, DC 20037; (202) 676-8928.

ICCAD-85, 1985 IEEE International Conference on Computer-Aided Design, November 11-14, Santa Clara, California. Contact Paul Weil, Hughes Aircraft (MS 270/055), B433 Fallbrook Dr., Canoga Park, CA 91034; (213) 702-1791.

Workstation 85, First International Conference and Exhibition on Computer Workstations, November 11-14, San Jose, California. Contact Edward Miller, Software Research Associates, 580 Market St., San Francisco, CA 94104; (415) 957-1441, or Workstation 85, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

microCOURSES

Microelectronic Circuit Packaging, Microprocessor Hardware, Software, and Interfacing, Pascal, Ada, and C in Microelectronic Design, Advanced Microprocessor System Design, courses held through July in various Canadian locations; $115-$475; in-house training available, **Contact: Ontario Centre for Microelectronics, Suite 400, 1150 Morrison Drive, Ottawa, Ontario K2H 9BB; (613) 596-6690.**

C Programming Workshop, Unix for Managers, Hands-On Unix for Programmers, courses held through June; $100-$750. **Contact: SSC, PO Box 7, Northgate Station, Seattle, WA 98125; (206) 367-8649.**

Occupational Ergonomics, June 3-7, 10-12; *Using the Ada Programming Language,* June 10-14; *Microcomputer Applications in Occupational Health and Safety Engineering,* June 17-18; *Database Management Using Personal Computers,* June 17-12; *Robotics: Concepts, Theory, and Applications,* June 24-28; *Principles of Microcomputers and Microprocessors,* June 15-19; *Written Communications for Engineers, Technical Writers, and Managers,* July 22-26; *Computer Vision and Image Processing,* July 29-August 2; *Contemporary Data Communication Networks,* August 5-9; $350-$975. **Contact: Engineering Summer Conferences, University of Michigan, 800 Chrysler Center, North Campus, Ann Arbor, MI 48109; (313) 764-8490.**

Data Communications System Components, Network Design, Operations, and Management, Network Protocols and Standards, SNA, LANs, Digital PBXs, courses held through July in various US locations; $750-$1350. **Contact: Systems Technology Forum, 9000 Fern Park Drive, Burke, VA 22015; (800) 336-7409.**

Micros for Managers: Software, eight videotape lectures available for lease or purchase. **Contact: Engineering Renewal and Growth, Colorado State University, Fort Collins, CO 80523; (800) 525-4950.**

Personal Computer and STD Computer Interfacing for Scientific Instrument Automation, August 22-24, Washington, D.C.; September 19-21, Greensboro, N.C.; $450. **Contact: C.E.C., Virginia Polytechnic Institute, Blacksburg, VA 24061; (703) 961-4848.**

Introduction to the Design of Fault-Tolerant Microcomputer Systems, May 6-8; $650. **Contact; University of Wisconsin—Extension, Dept. of Engineering and Applied Science, 432 North Lake Street, Madison, WI 53706; (800) 262-6243.**

Database Management and Fourth Generation Languages for Personal Computers, Integrated Software for the IBM PC, dBase II, dBase III, Introduction to Unix, Data Communications and Networking for the IBM PC and Other Personal Computers, courses held through June in various US locations; $695-895. **Contact: Software Institute of America, 8 Windsor Street, Andover, MA 01810; (617) 470-3880.**

ACCESS
recent books and articles on microcomputing

by Peter R. Rony
Dept. of Chemical Engineering
University of Delaware
Newark, DE 19716

Articles

Business Computer Systems

Vol. 3, No. 12, Dec. 1984:

H. Swartz, "The Case for Reverse Engineering," pp. 23-25.

E. Horwitt, "Long-Distance Networks," pp. 36-44.

L. Li, "Giving Voice to Computers," pp. 55-58.

J. Walden, "The Template Question," pp. 62-64.

I. H. Krakow, "Rigid Outline Structure Hampers Integrated Program," pp. 93-98.

"Business Computer Products: Summary Evaluations," pp. 117-124.

"Business Computer Products: An Exclusive Survey," p. 120.

Vol. 4, No. 1, Jan. 1985:

J. Martin, "We're Managing a Revolution . . . ," pp. 23-25.

H. Swartz, "On-Line Databases: The Legal Dilemma," pp. 28-29.

A. Bernstein, "Fit to Print," pp. 48-55.

S. Austin, "Multiuser Systems: Economies of Scale," pp. 67-86.

R. Lipton, "Business Computer Products: PC/Focus Brings Mainframe DBMS Power to Micros," pp. 91-98.

S.W. Bryan, "StatproXT Brings Serious Statistical Tools to Micros," pp. 98-104.

R. Glidewell, "The AT&T 3B2/300: Compact Unix Powerhouse," pp. 113-120.

Vol. 4, No. 2, Feb. 1985:

J. Walden, "Site Licensing," pp. 11-12.

D. W. Post, "The Fate of Unix," pp. 42-51.

E. Horwitt, "The Great LAN Software Breakthrough," pp. 54-57.

K. Mayo, "Business Battles Its In-House Pirates," pp. 60-65.

A. Bernstein, "Bar Codes Earn Their STRIPES," pp. 68-74.

S. Austin, "Feature Attractions," pp. 79-96.

I. H. Krakow, "Goldengate: Integration Plus a Micro-to-Mainframe Link," pp. 99-104.

R. W. Ridington, Jr., "Menlo's In-Search Simplifies Dialog Access, Saves Money," pp. 105-110.

Byte

Vol. 9, No. 13, Dec. 1984:

M. Vose, "The Tandy 1000," pp. 97-104.

S. Ciarcia, "Build the Power I/O System," pp. 105-118.

G. M. Vose, "C-Language: Development Tools," pp. 119-120, 382-386.

R. S. Shuford, "An Introduction to Fiber Optics: Part 1," pp. 121-123, 388-392.

G. Williams, "Software Frameworks," pp. 124-127, 394-410.

Special section on communications. Articles include "The Evolution of a Standard Ethernet," "Local-Area Networks for the IBM PC," "High-Speed Dial-Up Modems," "Writing Communications in Basic," and "Looking for the Perfect Program," pp. 131-215.

S. Lisanti, "The On-Line Search," pp. 215-230.

Byte reviews. Articles include "Reviewer's Notebook," "The Tandy Model 2000," "The Zenith Z-150 PC," "TK!Solver," "WordPerfect," "The Epson LQ-1500," "Review Feedback," pp. 237-301.

"*Byte* Guide to the Apple Personal Computer," pp. A4-A133.

G. Williams and R. Moore, "The Apple Story: Part 1—Early History," pp. A67-A71.

Vol. 10, No. 1, Jan. 1985:

B. Benson, "The Visual Mind and the Macintosh," pp. 113-130.

J. Nadan, "A Glimpse Into Future Television," pp. 135-150.

G. Williams, "Microsoft Macintosh Basic Version 2.0," pp. 155-162.

Books and reports

Bipolar LSI 1984 Databook, 5th ed., Monolithic Memories, Inc., 1984, free of charge.

Dataplot—Introduction and Overview, Cat. No. PB 84-214055, National Technical Information Service, 1984, $13.00 paper, $4.50 microfiche.

Directory of Graphics and CAD/CAM Packages, Report 45-BV, Data Decisions, 1985, $29.00.

Industrial Research Laboratories of the United States, 19th ed., R. R. Bowker Co., 1985, $145.00.

G. Kearsley and M. Furlong, *Computers for Kids Over 60,* Addison-Wesley Publishing Co., Inc., 1984, $9.95.

S. Lewis, *Plugging In: The Microcomputerist's Guide to Telecommunications,* Chilton Book Co., 1985, $11.95.

A Survey of Mathematical, Statistical, and Scientific Software Packages, Report 46-CD, Data Decisions, 1985, $29.00.

Systems Design Handbook, Monolithic Memories, Inc., 1983, free of charge.

Technology Trends in High-Performance Workstations, Report No. 1217, Strategic Inc., 1985, $1950.00.

G. B. Williams, *How to Repair and Maintain Your IBM PC,* Chilton Book Co., 1985, $12.95.

B. C. Yeung, *8086/8088 Assembly Language Programming,* John Wiley & Sons, Inc., 1984, $19.95.

G. Williams and R. Moore, "The Apple Story: Part 2—More History and the Apple III," pp. 167-180.

W. Rynone, "Uninterruptible Power Supplies," pp. 183-192.

R. S. Shuford, "An Introduction to Fiber Optics: Part 2—Connections and Networks," pp. 197-209.

P. A. Nilson, "Algorithms for a Variable-Precision Calculator," pp. 211-220.

C. Bigelow, "Font Design for Personal Workstations," pp. 255-270.

B. D'Ambrosio, "Expert Systems—Myth or Reality?" pp. 275-282.

Byte reviews. Articles include "Reviewer's Notebook," "The HP 110 Portable Computer," "Gifford's MP/M 8-16," "Lotus Symphony," "Magic Print," "The Hewlett-Packard ThinkJet Printer," "The TI Omni 800/Model 855 Printer," and "Review Feedback," pp. 289-353.

Vol. 10, No. 2, Feb. 1985:

P. Robinson, "The HP Integral Personal Computer," pp. 98-103.

S. Ciarcia, "Build a Serial EPROM Programmer," pp. 104-118.

J. Markoff and P. Robinson, "The Macintosh Office," pp. 120-134.

T. Carnevale, "C to Pascal," pp. 138-144.

J. L. Star, "Introduction to IMAGE Processing," pp. 163-170.

Special section on computing and the sciences. Articles include "The Birth of a Computer," "A Low-Cost Data-Acquisition System," "Fourier Smoothing Without the Fast Fourier Transform," "Paranoia: A Floating-Point Benchmark," "Modeling Mass-Action Kinetics," "Viewing Molecules With the Macintosh," "Laboratory Interfacing," and "Interfacing for Data Acquisition," pp. 177-269.

Byte reviews. Articles include "Reviewer's Notebook," "NewWord," "Janus/Ada," "The Epson Geneva PX-8," "Two Modula-2 Compilers for the IBM PC," "E-Mail for the Masses," "Mannesmann Tally MT 160," and "Review Feedback," pp. 289-331.

Computer

Vol. 17, No. 12, Dec. 198:

K. Kobori et al., "A 3-D CAD/CAM System With Interactive Simulation Facilities," pp. 14-21.

D. Laurent and S. Motet, "Geomatic: A 3-D Graphic Relief Simulation System," pp. 25-30.

H. Yoshiura, K. Fujimura, and T. L. Kunii, "Top-Down Construction of 3-D Mechanical Object Shapes From Engineering Drawings," pp. 32-40.

T. C. Woo, "Interfacing Solid Modeling to CAD and CAM: Data Structures and Algorithms for Decomposing a Solid," pp. 44-49.

Vol. 18, No. 1, Jan. 1985:

J. K. Huang, "The Input and Output of Chinese and Japanese Characters," pp. 18-24.

J. D. Becker, "Typing Chinese, Japanese, and Korean," pp. 27-34.

R. Matsuda, "Processing Information in Japanese," pp. 37-45.

H. Makino, "Beta: An Automatic Kana-Kanji Translation System," pp. 46-52.

W. Cui, "Evaluation of Chinese Character Keyboards," pp. 54-59.

J. Sheng, "A Pinyin Keyboard for Inputting Chinese Characters," pp. 60-63.

H. C. Tien, "The Pinxxiee Chinese Word Processor," pp. 65-66.

Computer Law & Practice

Vol. 1, No. 1, Sept./Oct. 1984:

D. G. Jerrard, "A Review of International Software Protection Measures," pp. 2-5.

J. Borking, "Patent Protection of Software in Continental Europe: A Status Quo," pp. 6-9.

A. Whitfield, "The Telecommunications Act of 1984," pp. 20-21.

S. Chalton, "Data Protection: New Civil Liberty or Heffalump Trap?" pp. 31-33.

Vol. 1, No. 2, Nov./Dec. 1984:

M. E. Beesley, "Regulation, Legislation, and the 1984 Telecommunications Act," pp. 38-41.

J. Borking, "Copyright Protection of Software in Contental Europe: A Status Quo," pp. 42-47.

D. Davies, "How to Avoid Being Bugged by Your Software: The Role for Insurance," pp. 50-51.

R. J. Hart, "Software Protection," pp. 62-64.

"Case Reports," pp. 67-70.

Datamation

Vol. 30, No. 20, Dec. 1, 1984:

R. E. Carlyle, "Vendor Outlook Bleak," pp. 36-38.

W. Schatz, "High Stakes at NSF," pp. 45-52.

W. Stallings, "The Integrated Services Digital Network," pp. 68-80.

"Systems Software Survey: Users' Favorite Disks," pp. 84-138.

Vol. 30, No. 21, Dec. 15, 1984:

D. S. Appleton, "The State of CIM," pp. 66-72.

Vol. 31, No. 1, Jan. 1, 1985:

N. Burnett and J. Neimark, "In Focus: DP and the Disabled," pp. 22-30.

R. J. Crutchfield, "Can MS/NET Succeed?" pp. 38-44.

W. Schatz, "Battle of the Boards," pp. 61-64.

Special section on IBM. Articles include "Fast Break in Armonk," "Shopping for Market Share," "The Users' Story," "IBM: Mainframes in 1990," "A Generous Portion," and "Banking on IBM," pp. 68-108.

Vol. 31, No. 2, Jan. 15, 1985:

P. H. Dorn, "Learning From Lemons," pp. 72-80.

E. Sigel, "Alas Poor VisiCorp," pp. 93-96.

C. Bassine, "The Computer Expert's Guide to Life," pp. 101-104.

D. M. Bowers, "The Importance of Power," pp. 116-122.

Vol. 31, No. 3, Feb. 1, 1985:

L. L. Baird, Jr., "In Focus: Sensible Network Security," pp. 22-28.

R. E. Carlyle, "DEC Puts VAX on Eight ICs," pp. 34-36.

L. O'Keeffe, "IBM's OA Puzzle," pp. 74-78.

L. D. Passmore, "The Networking Standards Collision," pp. 98-108.

D. V. Morland, "The Evolution of Software Architecture," pp. 123-132.

Vol. 31, No. 4, Feb. 15, 1985:

R. E. Carlyle, "In Focus: RISC-Y Business," pp. 30-36.

D. Stamps, "Bean Counters Attack!" pp. 64-76.

J. W. Verity, "Bridging the Software Gap," pp. 84-88.

D. Dee, "Developing PC Applications," pp. 112-116.

P. J. Stevenson, "A New Voyage for Columbus," pp. 134-138.

Dr. Dobb's Journal

Vol. 9, No. 12, Dec. 1984:

Special section on Unix. Articles include "Varieties of Unix," "Unix Device Drivers," and "A Unix Internals Bibiliography," pp. 24-59.

J. R. Johnson, "A File Browser Program," pp. 60-76.

H. A. Seymour, "An Introduction to Parsing," pp. 78-86.

Vol. 10, No. 1, Jan. 1985:

T. Lafleur and S. Raab, "Fatten Your Mac," pp. 18-23.

T. Mayer, "QuickDraw Meets Imagewriter," pp. 26-35.

I. E. Ashdown, "Archiving Files With CP/M-80 and CP/M-86," pp. 36-60.

R. Wilton, "Unstructured Forth Programming," pp. 86-87.

"Software Reviews: APL*PLUS/PC, Version 3.1; Tools, Volume 1; and Financial and Statistical Library," pp. 88-102.

"VSI—Virtual Screen Interface, Version 2.09," pp. 102-103.

"The Fancy Font System, Version 2.0," pp. 103-105.

R. Duncan, "16-Bit Software Toolbox," pp. 108-122.

Vol. 10, No. 2, Feb. 1985:

100th issue of *Dr. Dobb's. IEEE Micro* extends its congratulations to People's Computer Company, the publisher (M&T Publishing, Inc.), the editors, and the contributors of one of the very special and valuable magazines in the microcomputer area.

S. Rodriguez, T. Pittman, and B. Albrecht, "Festschrift for Doctor Dobb," pp. 26-30.

P. Freiberger and M. Swaine, "Fire in the Valley," pp. 32-40.

G. Brandly, "Tiny Basic for the 68000," pp. 42-46.

C. E. Burton, "An Enhanced ADFGVX Cipher System," pp. 48-70.

G. Head, "More dBase Tips and Techniques," pp. 71-73.

"Reviews: Modula-2/86, Version 1.04," pp. 74-93.

R. Blum, "CP/M Exchange," pp. 98-119.

Electronics Week

Vol. 57, No. 34, Dec. 3, 1984:

T. Manuel, "Cautiously Optimistic Tone Set for 5th Generation," pp. 57-63.

J. Crowley, "Automation Smooths CAD Process," pp. 65-69.

Vol. 57, No. 35, Dec. 10, 1984:

E. J. Lerner, "ICs Nudging the Submicron Geometry Limit," pp. 51-58.

R. Valentine and J. Stewart, "Automakers Shift to Processors," pp. 61-65.

Vol. 57, No. 36, Dec. 17, 1984:

W. R. Iversen, "VHSIC—Insertion Program Begins to Pay Dividends," pp. 57-66.

P. M. Shaw, Jr., "How Copyright Act Protects IC Masks," pp. 99-102.

R. A. Sehr, "Two Low-Speed Nets Race to Link Computers," pp. 107-110.

Vol. 58, No. 1, Jan. 1, 1985:

C. L. Cohen, "CMOS Chips Make a 'Hybrid'," pp. 17-18.

K. A. Marks, "MOS Meets Radiation Challenge," pp. 89-91.

Vol. 58, No. 2, Jan. 7, 1985:

A. Porter, "Sampling Sees Skinny Signals," pp. 41-44.

Vol. 58, No. 3, Jan. 14, 1985:

R. Lineback, "DRAM Makers Gird for 256-K," pp. 13-17.

R. Rosenberg, "Is the VMEbus Losing Its Lead?" pp. 30-31.

C. Barney, "Award for Achievement," pp. 40-44.

Vol. 58, No. 4, Jan. 21, 1985:

T. Feldt, "Satellites—The Push Is On," pp. 45-49.

Vol. 58, No. 5, Feb. 4, 1985:

D. M. Stewart, "Lasers Fix Dynamic RAMs," pp. 45-49.

T. Manuel and M. B. Rand, "Has AI's Time Come at Last?" pp. 51-62.

E. Macaruso, "DOD Vexes IC Makers," pp. 63-68.

J. R. Lineback, "Civilian Chips Don Uniforms," pp. 72-74.

Vol. 58, No. 6, Feb. 11, 1985:

R. Rosenberg, "Super Cube," pp. 15-17.

J. R. Lineback, "DSP Chips Spark Lackluster Market," pp. 20-21.

C. Barney, "Chips Forging Telecom Links," pp. 26-27.

B. C. Cole, "Memories Dominate ISSCC," pp. 51-60.

R. Schlater, "Digital Scopes Gain Persistence," pp. 61-65.

S. Zollo, "IC Keeps Pirates Out," pp. 73-76.

IEEE Circuits and Devices Magazine

Vol. 1, No. 1, Jan. 1985:

A. J. Kessler and A. Ganesan, "Standard Cell VLSI Design: A Tutorial," pp. 17-34.

M. P. Lepselter and S. M. Sze, "DRAM Pricing Trends—The π Rule," pp. 53-54.

IEEE Communications Magazine

Vol. 22, No. 12, Dec. 1984:

S. Lin, D. J. Costello, Jr., and M. J. Miller, "Automatic-Repeat-Request Error-Control Schemes," pp. 5-17.

V. I. Johannes, "Improving on Bit Error Rate," pp. 18-20.

W. Stallings, "Digital Signaling Techniques," pp. 21-25.

J. L. Massey, "Information Theory: The Copernican System of Communications," pp. 26-28.

Vol. 23, No. 1, Jan. 1985:

Special issue on telecommunications standards.

Vol. 23, No. 2, Feb. 1985:

J. B. H. Peek, "Communications Aspects of the Compact Disc Digital Audio System," pp. 7-15.

J. D. Crow, "Computer Applications for Fiber Optics," pp. 16-20.

K. Kobayashi, "Advances in Facsimile Art," pp. 27-35.

IEEE Computer Graphics and Applications

Vol. 4, No. 10, Oct. 1984:

L. Lichten, "Computer-Aided Design Applications on Microcomputers," pp. 25-28.

Vol. 5, No. 2, Feb. 1985:

L. G. Richards, "Engineering Education: A Status Report on the CAD/CAM Revolution," pp. 19-25.

IEEE Spectrum

Vol. 22, No. 1, Jan. 1985:

"Best Bits: Applications of Microprocessors," p. 30.

Special issue entitled "Technology '85." Articles include "The Experts' Outlook," "Minis and Mainframes," "Personal Computers," "Software," "Microprocessors," "Communications," "Solid State," "Instrumentation," "Industrial Electronics," "Power and Energy," "Consumer Electronics," "Transportation," "Aerospace and Military," and "Medical Electronics," pp. 34-96.

Vol. 22, No. 2, Feb. 1985

M. F. Leahy, "Superfine IC Geometries," pp. 36-43.

G. Borriello, R. H. Katz, A. G. Bell, and L. Conway, "VLSI System Design by the Numbers," pp. 44-50.

Vol. 22, No. 3, Mar. 1985:

"Best Bits: Applications of Microprocessors," p. 34.

T. S. Perry and P. Wallich, "Design Case History: The Commodore 64," pp. 48-58.

F. P. Kapron, "Fiber-Optic System Tradeoffs," pp. 68-75.

IEEE Transactions on Industrial Electronics

Vol. IE-32, No. 1, Feb. 1985:

P. Chaudhuri, K. Ray, and S. Ghosh, "A Real-Time Process Scheduler for a Ring-Type Microcomputer Network," pp. 56-61.

A. Bellini, G. Figalli, and G. Ulivi, "A High-Performance Microcomputer-Based Control Circuit for Variable Frequency Inverters," pp. 62-70.

E. E. Mitchell and R. Demoyer, Jr., "A Versatile Digital Controller Algorithm Incorporating a State Observer and State Feedback," pp. 78-84.

Infosystems

Vol. 31, No. 12, Dec. 1984:

C. A. Krislov, "Getting Uncle Sam to Help Pay for Your Software Development Costs: Part II," pp. 86-87.

Vol. 32, No. 2, Feb. 1985:

M. V. Janulaitis, "Creating a Disaster Recovery Plan," pp. 42-43.

M. Walsh, "Unscrambling the Kaleidoscope: A Look at IBM's PCs," pp. 46-52.

J. Snyder, "Semi-Annual Software Review: The New Wave of Software Vendors," pp. 62-85.

Macworld

Vol. 2, No. 1, Jan. 1985:

D. Goodman, "A Shopper's Guide to 4*Software," pp. 38-44.

N. McGoldrick, "Filevision: A Data Base in Pictures," pp. 62-71.

J. Heid, "Covering All the Bases," pp. 78-91.

G. McComb, "Making the Most of the Mac's Fonts," pp. 106-119.

Vol. 2, No. 2, Feb. 1985:

J. Felici and E. Spire, "A Face for All Seasons," pp. 44-50.

D. Goodman, "The Laser's Edge," pp. 70-79.

B. Grout, "Just the Facts," pp. 84-88.

R. Sprague, "The Language That Talks to Your Printer," pp. 106-115.

B. Grout, "The Elements of Graphic Design," pp. 122-129.

Microprocessors and Microsystems

Vol. 8, No. 8, Oct. 1984:

O. R. Omotayo, "A Talking Prestel Terminal," pp. 403-412.

S. M. Said and K. R. Dimond, "Design Considerations for a Hardware-Refreshed Memory Card for the MC68000," pp. 413-416.

M. J. Taylor, "Design of Externally Triggered Data Acquisition Units," pp. 417-423.

C. Crook, "The Digital Storage Oscilloscope (DSO) as a Cost-Effective Solution to Measurements in Digital Circuits," pp. 435-439.

Vol. 8, No. 9, Nov. 1984:

P. R. Roper, "68000 PDOS—An Overview," pp. 458-469.

M. P. Reddy and B. S. Bhanumurthy, "Microprocessor-Based PCM Decommutator," pp. 470-474.

P. K. Dash and D. K. Panda, "Spectral Observation of Power Network Signals for Digital Signal Processing," pp. 475-480.

O. R. Omotayo, "Converting Text Into Speech in Real Time With Microcomputers," pp. 481-487.

A. A. Wahab, R. Nagarajan, and D. H. Jerew, "Database Management in a Microcomputer-Controlled IC Tester," pp. 488-491.

T. V. Karthikeyan, K. S. Rajashekara, and S. R. Bhat, "Microprocessor-Based Event Sequence Recorder," pp. 492-497.

J. F. Poiraudeau and S. Roux, "PL/M Codes Mixing on Single-Board Systems," pp. 498-500.

Mini-Micro Systems

Vol. 17, No. 15, Dec. 1984:

M. Stenzler-Centonze, "Ungermann-Bass, GE Team Up to Connect Factory Systems," pp. 35-37.

L. Haber, "Graphical Kernel System (GKS) Makes Its Mark in Software Markets," pp. 97-102.

T. A. Oliver, "Embedded Servo Controllers Push Up Disk Storage," pp. 117-122.

D. Collier, P. D. Frank, and C. J. Aho, "Rigid Disk Heads Keep Pace With Growing Storage Needs," pp. 127-133.

J. F. Ready, "Operating Systems Conform to Application Needs," pp. 137-143.

D. P. Misunas, "Protocol Converters Link Incompatible Devices," pp. 147-159.

L. Harris, "Teaching Computers English Proves Easier Than Training People," pp. 163-172.

R. Dalrymple, "Multiprocessor Architectures Spark Interest in 32-Bit Buses," pp. 177-184.

Vol. 18, No. 1, Jan. 1985:

R. Dalrymple, "Printer Manufacturers Elbow for Shelf Space," pp. 128-136.

C. Warren, "Word-Processing Software Injects Text Into Pictures," pp. 143-154.

C. Warren, "High-Capacity 8-Inch Winchesters Aim to Overtake 14-Inch Market," pp. 185-191.

J. Estrin, "Hybrid Technologies Rewrite the Rules for Local-Area Networks," pp. 195-204.

Vol. 18, No. 2, Feb. 1985:

D. W. Basehore and H. B. Hazebrouck, "Half-Height Drive Packs 70M-Byte Power," pp. 111-118.

C. Warren, "SCSI Bus Eases Device Integration," pp. 123-131.

I. D. Allan, "Varied Drive Interfaces Mystify Integrators," pp. 135-144.

M. Tucker, "Unix Emerges as a Universal Tool Kit," pp. 149-175.

Mundo Electrónico

No. 143, Sept. 1984:

F. S. Alonso, "Normalización del Lenguaje Basic," pp. 87-93.

M. C. R. de Tejada, "Introducción a la Ingeniería del Software," pp. 95-107.

J. P. Franco, "Simulación Asistida por Ordenador," pp. 109-114.

No. 144, Oct. 1984:

Special issue entitled "La Voz de la Electrónica."

M. Rodriguez et al., "Visión Panorámica de la Respuesta Oral de Maquinas," pp. 57-66.

M. Rodriguez et al., "Alternativas Para Síntesis de Voz," pp. 67-79.

A. G. Sanchez, "Modelos Para Reconocimiento de Palabras con Independencia del Locutor," pp. 81-88.

S. Aguilera et al., "Obtención-Visualización de Algunos Parámetros del Habla," pp. 89-93.

E. Vidal, H. Rulot, and F. Casacuberta, "Reconocimiento de Palabras Aisladas Mediante μP," pp. 95-102.

J. Meneses et al., "Arquitecturas Digitales LSI-VLSI Para el Procesamiento del Habla," pp. 103-108.

R. G. Gomez et al., "Codificación Predictiva de la Senal de Voz," pp. 111-116.

R. Inigo and J. M. Angulo, "Visión por Computador y Su Aplicación a la Robotica," pp. 151-157.

J. A. Miranda and I. F. Crespo, "Buses Normalizados Para Tarjetas μP," pp. 184-196.

PC Tech Journal

Vol. 3, No. 1, Jan. 1985:

R. Larson, "Snobol4," pp. 32-43.

P. Finan, "PACKing the Date and Time," pp. 46-54.

J. Anderson, "The Maverick Controller," pp. 58-63.

C. Christian, "Take III," pp. 64-84.

S. Armbrust and T. Forgeron, "Entymological Explorations," pp. 88-105.

A. Hansen, "Kermit," pp. 110-123.

A. Hansen, "Telios," pp. 130-139.

J. Creane, "The Limited Joys of Translated Software," pp. 142-160.

M. A. Covington, "Documentation That Works," pp. 165-172.

Vol. 3, No. 2, Feb. 1985:

T. Mirecki, "Dipping Into Directories," pp. 67-77.

A. Chaturvedi, "Tree Structures," pp. 78-87.

R. B. Stam, "Environmental Excavations," pp. 90-98.

J. Duntemann, "Tools for the Pascal Programmer," pp. 102-111.

M. Covington, "The Power of Turbo Pascal," pp. 112-123.

J. S. Mallozzi, "Language Learning Tools," pp. 126-139.

A. Hansen, "Electronic Messaging," pp. 141-149.

M. Abrash and D. Illowsky, "Customized Boots," pp. 150-159.

PC World

Vol. 3, No. 1, Jan. 1985:

Special issue entitled "The Creative Side of Word Processing."

J. Martin, "New Dimensions in Word Processing," pp. 42-51.

M. Shapiro, "Dialing for Dividends," pp. 68-78.

E. Brown, "Word Processing '85," pp. 100-105.

E. Wallace, "THOR: New Thunder," pp. 108-117.

M. Shinyeda, "WordMARC: An Office Powerhouse," pp. 126-133.

D. D. Beckman, "DisplayWrite 2: In the Corporate Mold," pp. 138-143.

P. Giese, "Due Processing," pp. 192-201.

Vol. 3, No. 2, Feb. 1985:

M. K. Guttman, "Strategies for Sharing Resources," pp. 42-50.

J. Sachs, "Six Leading LANs," pp. 108-128.

A. Wilcox, "Untangling Networks," pp. 232-241.

Vol. 3, No. 3, Mar. 1985:

M. Shinyeda, "Word Processing: The Deciding Factors," pp. 52-57.

C. Humble, "Non-Taxing Software," pp. 122-127.

A. Pilgrim, "Bar Code Bonanza," pp. 198-207.

Systems and Software

Vol. 3, No. 12, Dec. 1984:

R. Bernhard, "XT 370 Pushes IBM Plan to Link All Products via VM/CMS," pp. 63-66.

R. Bernhard, "RISCs—Reduced-Instruction-Set Computers—Make Leap," pp. 81-84.

J. McLeod, "Optical Disks: Mass Storage of Information," pp. 102-115.

K. Mankin, "Supermicro Runs MS-DOS Under Unix," pp. 116-121.

M. Freeman and C. Kaplinsky, "Coprocessors Up System Throughput," pp. 122-130.

U. T. Murty et al., "3B Computers Can Talk to Diversity of Systems," pp. 131-135.

M. Szabados, "Network Software Builds on ISO Standard," pp. 137-141.

J. P. McNaul, "Pictures Merged With Words," pp. 143-146.

B. Metcalfe and D. Vaskevitch, "Multiuser PC Networks Versus Timesharing," pp. 147-158.

Vol. 4, No. 1, Jan. 1985:

W. Rauch-Hindin, "IBM Plays 'Shell' Game With PC-DOS," pp. 31-34.

D. R. Brousell, "While Microsoft Ships Networks, Theories About IBM Persist," pp. 35-38.

S. Feldman, "DEC Debuts 8600, But Can It Move Past Installed Base?" pp. 42-52.

D. R. Brousell, "How Key Operating Systems Will Evolve in '85," pp. 109-112.

W. Rauch-Hindin, "AI Technology Ready for Commercialization," pp. 115-122.

R. Bernhard, "The Fifth Generation—Awesome Obstacles," pp. 132-138.

J. Malpas, "Logic Programming, Through Prolog, Steps Into the Real World," pp. 147-152.

A. Lim, "32-Bit Workstations Standardize Systems," pp. 165-170.

Vol. 4, No. 2, Feb. 1985:

E. L. Keller, "Real Time Becomes Commercial Issue," pp. 86-89.

"Mini Manufacturers See Real Time for 16- and 32-Bit Machines," pp. 92-94.

"Unix Compatibility Becomes Real-Time Issue for Vendors," pp. 94-102.

"Vendors Are Doing Time With Real-Time Kernels," pp. 102-106.

R. E. Peterson, Jr., "Packing More Power per Board," pp. 109-118.

G. W. Everhart, "Net Links IBM PCs Directly to IBM Hosts," pp. 119-121.

P. D. MacWilliams, "Multiple Buses for Future Systems," pp. 123-127.

S. P. Harbison, "Performance, Productivity, and Your C Compiler," pp. 129-133.

Addresses of publishers

Addison-Wesley Publishing Co., Inc.
Jacob Way
Reading, MA 01867
(617) 944-3700

R. R. Bowker Co.
205 E. 42nd St.
New York, NY 10017
(212) 916-1600

Business Computer Systems
Cahners Publishing Co.
270 St. Paul St.
Denver, CO 80206
(303) 388-4511
(Publisher does not sell back issues or single copies of articles. Subscriptions are free to qualified readers.)

Byte
Byte Publications Inc.
70 Main St.
Peterborough, NH 03458
(603) 924-9281

Chilton Book Co.
Chilton Way
Radnor, PA 19089
(800) 345-1214,
in PA (215) 964-4729

Computer
IEEE Computer Society
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720
(714) 821-8380

Computer Design
Computer Design Publishing Co.
11 Goldsmith St.
Littleton, MA 01460
(617) 486-8944

Computer Law & Practice
Frank Cass & Co., Ltd.
Gainsborough House
11 Gainsborough Rd.
London E11 1RS UK

Data Decisions
20 Brace Rd.
Cherry Hill, NJ 08034
(609) 429-7100

Datamation
Technical Publishing Co.
1301 S. Grove Ave.
Barrington, IL 60010
(312) 774-8115

Dr. Dobb's Journal
People's Computer Co.
2464 Embarcadero Way
Palo Alto, CA 94304
(415) 424-0600

Electronics Week
McGraw-Hill Publications Co.
1221 Ave. of the Americas
New York, NY 10020
(212) 512-2000

IEEE Circuits and Devices Magazine
IEEE Communications Magazine
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854
(201) 981-0060, x133

IEEE Computer Graphics and Applications
IEEE Computer Society
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720
(714) 821-8380

IEEE Spectrum
IEEE Transactions (all)
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854
(201) 981-0060, x133

Infosystems
Hitchcock Publishing Co.
Hitchcock Bldg.
23 W. 550 Geneva Rd.
Wheaton, IL 60187
(312) 665-1000

Macworld
555 De Haro St.
San Francisco, CA 94107
(415) 861-3861

Microprocessors and Microsystems
Business Press International
205 E. 42nd St., Suite 1705
New York, NY 10017
(212) 867-2080

or

Butterworth Scientific Ltd.
Journals Division
PO Box 63
Westbury House
Bury St.
Guildford, Surrey GU2 5BH UK

Mini-Micro Systems
Cahners Publishing Co.
270 St. Paul St.
Denver, CO 80206
(303) 388-4511

Monolithic Memories, Inc.
2175 Mission College Blvd.
Santa Clara, CA 95054-1592
(408) 970-9700

Mundo Electrónico
Boixareau Editores, SA
Grand Via de les Corts Catalanes 594
2 Barcelona 2 Spain

National Technical Information Service
US Dept. of Commerce
5285 Port Royal Rd.
Springfield, VA 22161
(703) 487-4650

PC Tech Journal
Ziff-Davis Publishing Co.
1 Park Ave.
New York, NY 10016
(212) 725-7947

PC World
555 De Haro St.
San Franciso, CA 94107
(415) 861-3861

Strategic Inc.
PO Box 2150
Cupertino, CA 95015-2150
(408) 446-4500

Systems and Software
Hayden Publishing Co., Inc.
50 Essex St.
Rochelle Park, NJ 07662
(201) 843-0550

John Wiley & Sons, Inc.
One Wiley Dr.
Somerset, NY 08873
(201) 469-4400

ADVERTISER/PRODUCT INDEX

April 1985

Advertisers

Products

FOR DISPLAY ADVERTISING INFORMATION CONTACT

Southern California and Mountain States: Richard C. Faust Company, 24050 Madison Street, Suite 100, Torrance, CA 90505; (213) 373-9604.
Northern California and Pacific Northwest: Don Farris Company, 161 W. 25th Ave., #102B San Mateo, CA 94403; (415) 349-2222.
Jack Vance, P.O. Box 3205, Saratoga, CA 95070, (408) 741-0354
New England: Arpin Associates, P.O. Box 227, Weston, MA 02193; (617) 899-5613.
George Watts, III, 4 Conifer Dr., Wilbraham, MA 01095; (413) 596-4747.
East Coast: Hart Associates, P.O. Box 339, 42 Lake Blvd., Matawan, NJ 07747; (201) 583-8500.
Midwest: Jerry Berk, Special Market Media, 421 Florence Ave., Evanston, IL 60202; (312) 271-2001.
Southeast: Larry C. Shattles, 133 Laurel Oak Drive, Longwood, FL 32779; (305) 788-1950.
Southwest: The House Company, 3817 Richmond Avenue, Suite 110, Houston, TX 77027; (713) 622-2868.

Advertising Manager: Mike Koehler, *IEEE MICRO* Magazine, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720 (714) 821-3240, 821-8380.

For production information, conference or classified advertising contact Sandra J. Arteaga, *IEEE MICRO* Magazine, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, (714) 821-1140.

Moving?

PLEASE NOTIFY
US 4 WEEKS
IN ADVANCE

MAIL TO:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

Name (Please Print)

New Address

City State/Country Zip

ATTACH LABEL HERE

- This notice of address change will apply to all publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

IEEE MICRO READER SERVICE

For further information on advertised products, new products, or literature, fill out one of these cards. Circle the number on the Reader Service Card that corresponds to the number at the bottom of the item in which you are interested. Affix stamp and mail.

Please print or type your name and address.

READER SERVICE CARD

IEEE MICRO

Void after October 15, 1985 **4/85**

Name ____________________

Company ____________________

Address ____________________

City ____________ State ________ Zip ________

Title ____________________

Telephone number ( ) ____________________

101 ☐ Please send me the IEEE-CS *Publications Catalog*.

102 ☐ Please send me an IEEE fellows nomination form.

Articles I liked in this issue:

A ○ Microview (p. 4)
B ○ Computer-aided Program. System (p. 9)
C ○ Electronic Scanners (p. 20)
D ○ Structure for Comp. Networking (p. 53)
E ○ "On the Fly" CRC-16 Calc. (p. 67)
F ○ Dint: Data Integration (p. 76)
G ○ Micronews (p. 83)
H ○ Microstandards (p. 86)
I ○ Microlaw (p. 88)
J ○ Microreview (p. 92)
K ○ New Products (p. 95)
L ○ Product Summary (p. 103)
M ○ Access (p. 106)

PRODUCTS PURCHASED OR SPECIFIED	FOR JOB	FOR HOBBY
Computer	☐	☐
Peripherals	☐	☐
Data communications	☐	☐
Memories, components	☐	☐
Software and services	☐	☐
Publications	☐	☐
Other	☐	☐

Send more information on numbered items:

1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65	69	73	77	81	85	89	93	97
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70	74	78	82	86	90	94	98
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75	79	83	87	91	95	99
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100

READER SERVICE CARD

IEEE MICRO

Void after October 15, 1985 **4/85**

Name ____________________

Company ____________________

Address ____________________

City ____________ State ________ Zip ________

Title ____________________

Telephone number ( ) ____________________

101 ☐ Please send me the IEEE-CS *Publications Catalog*

102 ☐ Please send me an IEEE fellows nomination form.

Articles I liked in this issue:

A ○ Microview (p. 4)
B ○ Computer-aided Program. System (p. 9)
C ○ Electronic Scanners (p. 20)
D ○ Structure for Comp. Networking (p. 53)
E ○ "On the Fly" CRC-16 Calc. (p. 67)
F ○ Dint: Data Integration (p. 76)
G ○ Micronews (p. 83)
H ○ Microstandards (p. 86)
I ○ Microlaw (p. 88)
J ○ Microreview (p. 92)
K ○ New Products (p. 95)
L ○ Product Summary (p. 103)
M ○ Access (p. 106)

PRODUCTS PURCHASED OR SPECIFIED	FOR JOB	FOR HOBBY
Computer	☐	☐
Peripherals	☐	☐
Data communications	☐	☐
Memories, components	☐	☐
Software and services	☐	☐
Publications	☐	☐
Other	☐	☐

Send more information on numbered items:

1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65	69	73	77	81	85	89	93	97
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70	74	78	82	86	90	94	98
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75	79	83	87	91	95	99
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100

Comments on articles and other editorial matter

I liked: ______________________________

____

____

____

____

I disliked: ______________________________

____

____

I would like: ______________________________

____

____

____

For reader service, see other side

This PO box for reader service cards only.

PLACE STAMP HERE

IEEE MICRO

Reader Service Inquiries
Box 24168
Los Angeles, CA 90024
USA

Comments on articles and other editorial matter

I liked: ______________________________

____

____

____

____

I disliked: ______________________________

____

____

I would like: ______________________________

____

____

____

For reader service, see other side

This PO box for reader service cards only.

PLACE STAMP HERE

IEEE MICRO

Reader Service Inquiries
Box 24168
Los Angeles, CA 90024
USA

IEEE COMPUTER SOCIETY
Membership Application

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Join now and receive one of these popular books absolutely free!

***Microprocessors and Microcomputers*—**

All-new third edition — Selected papers from *Computer* and *IEEE Micro* (372 pp.)

Regular nonmember price: $20.00

***Selected Reprints in Software*—**

Selected papers from *Computer* (282 pp.)

Regular nonmember price: $12.00

Offer does not apply to student member applicants.

Check here for your free book. Check one:

☐ *Microprocessors and Microcomputers*

or

☐ *Selected Reprints in Software*

Check box for additional information:

☐ Computer Society Membership Brochure
☐ Bulk air option (two-to-six-week delivery of publications) for members in Western Europe, Latin America, and the Far East
☐ Student Membership Application
☐ Publications Catalog

Schedule of Fees (tax deductible)

IMPORTANT: Pay the FULL-YEAR rate if your application is postmarked between September 1, 1984 and February 28, 1985. Pay the HALF-YEAR rate if your application is postmarked between March 1, 1985 and August 31, 1985. Membership expires December 31, 1985.

For applications received:	Half-Year 3/1/85–8/31/85	Full-Year 9/1/84–2/28/85
• **an IEEE member who wishes to join the Computer Society**	☐ **$5.00**	☐ **$10.00**
IEEE membership no. ______		
• **a non-IEEE member who wishes to join both IEEE and the Computer Society*** Full IEEE members receive *Spectrum*, *The Institute*, and many additional benefits.	☐ **$52.50**	☐ **$90.00**
• **a non-IEEE member who wishes to join the Computer Society only**	☐ **$17.00**	☐ **$34.00**

*Individuals who join both the IEEE and the Computer Society in Regions 7 (Canada) and 8 (Europe, Mid-East, & Africa) may deduct $6 from full-year rates/$3 from half-year rates; in Region 9 (Latin America), deduct $13/$6.50; in Region 10 (Asia & Pacific), deduct $12/$6.

ACM members who join both the IEEE and the Computer Society may deduct $5 from full-year rates/$2.50 from half-year rates.

Publication Options

If, in addition to your automatic subscription to *Computer* Magazine, you would like to receive:

add:	Half-Year	Full-Year
• *IEEE Computer Graphics & Applications* (3061)	☐ $11.00	☐ $22.00
• *IEEE Micro* (3071)	☐ $7.50	☐ $15.00
• *IEEE Design & Test of Computers* (3111)	☐ $6.00	☐ $12.00
• *IEEE Software* (3121)	☐ $7.00	☐ $14.00
• *Transactions on Computers* (1161)	☐ $7.50	☐ $15.00
• *Transactions on Software Engineering* (1171)	☐ $8.00	☐ $16.00
• *Transactions on Pattern Analysis & Machine Intelligence* (1351)	☐ $6.00	☐ $12.00
• *Journal on Lightwave Technology* (4301)	☐ $5.00	☐ $10.00
• *Journal of Robotics and Automation* (4401)	☐ $5.00	☐ $10.00
• *Journal of Solid State Circuits* (4101)	☐ $2.50	☐ $5.00
• *Proceedings of the IEEE* (5011) (available only to IEEE members)	☐ $7.50	☐ $15.00

Total amount remitted with this application $______

☐ Check or money order enclosed. (Make payable to IEEE. Mail to IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.)

☐ VISA ☐ MasterCard ☐ American Express

Charge Card Number ______ Exp. Date Mo. | Yr.

I hereby make application for IEEE Computer Society membership and, if elected, will be governed by IEEE's and the Society's Constitutions, Bylaws, and Statements of Policies and Procedures.

Full signature ______ Date ______

MAILING ADDRESS

First Name ______ Middle initial(s) ______ Last name ______

Street address ______

City ______ State/Country ______ Zip ______

EDUCATION (highest level completed) Course ______ Degree received ______ Date ______

Name of educational institution ______ Date of birth ______ ☐ Male ☐ Female

OCCUPATION Title or position ______

Firm name ______

Firm address ______

City ______ State/Country ______ Zip ______

Affiliate society membership no. (if applicable) ______ Grade ______

REFERENCE (an IEEE member; if unknown, a managerial person who knows you professionally.)

Name (print in full) ______

Position ______

Street address ______

City ______ State/Country ______ Zip ______

MAIL TO: IEEE COMPUTER SOCIETY, 10662 LOS VAQUEROS CIRCLE, LOS ALAMITOS, CA 90720.

PMCRO 485

CALL FOR PAPERS

IEEE MICRO

IEEE Micro Special issue on engineering workstations

The October 1985 issue of ***IEEE Micro*** will examine the use of computer workstations in all areas of engineering. Emphasis will be on user requirements, with equal attention given to industry and education. Practical workstation applications will be of particular interest.

Submissions should be sent by June 10 to:

Dr. Donald O. Knight
Engineering Research Center
Mail Stop ECG
Arizona State University
Tempe, AZ 85281

Please submit six copies of your manuscript.

For further information about the special issue, contact Dr. Knight at (602) 965-5346.